UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RICARDO JOSÉ PFITSCHER

# Monitoring and Identifying Bottlenecks in Virtual Network Functions Service Chains

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Lisandro Zambenedetti
Granville

Porto Alegre
Janeiro 2019

**CIP – CATALOGING-IN-PUBLICATION**

Pfitscher, Ricardo José

Monitoring and Identifying Bottlenecks in Virtual Network Functions Service Chains / Ricardo José Pfitscher. – Porto Alegre: PPGC da UFRGS, 2019.

122 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Advisor: Lisandro Zambenedetti Granville.

1. Network Functions Virtualization. 2. Bottlenecks Identification. 3. Performance Analysis. I. Granville, Lisandro Zambenedetti. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitor: Prof. Jane Fraga Tutikian
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretor do Instituto de Informática: Prof. Carla Maria Dal Sasso Freitas
Coordenador do PPGC: Prof. João Luiz Dihl Comba
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

*“Action may not always bring happiness;*
*but there is no happiness without action”*
— Benjamin Disraeli

## ACKNOWLEDGEMENT

Acknowledgments are written in Portuguese because it is my natural language, and to allow all members of my family to read it.

O desenvolvimento de um doutorado contempla, além dos benefícios diretos oriundos de se conquistá-lo, uma série de percalços que necessitam ser atravessados; de modo algum é possível enfrentar este desafio sozinho. Assim, quero agradecer todos aqueles que de alguma forma me ajudaram durante o desenvolvimento desta etapa tão importante em minha vida. Por isso, obrigado, e desde-já peço desculpas se porventura me esquecer de alguém.

A escolha pela vida acadêmica teve participação muito importante dos professores Rafael Rodrigues Obelheiro e Maurício Aronne Pillon, meus "gurus" acadêmicos. Ambos me guiaram desde a graduação até a conclusão do mestrado, e por incentivo e ajuda deles me encaminhei ao doutorado. Naquela época eu não entendia direito o papel de ser pesquisador, tão pouco, conhecia a minha paixão pela docência. O incentivo e ensinamento de ambos foram fundamentais para a escolha desta trajetória.

Em continuidade, os professores do instituto de informática da UFRGS tiveram grande valia durante o processo. Os ensinamentos obtidos nas disciplinas cursadas ajudaram, certamente, a fundamentar as teorias envolvidas na construção desta tese. Neste sentido, vale ressaltar a participação imprescindível do meu orientador, Prof. Lisandro, indicando as direções corretas para encontrar as dúvidas que eu deveria sanar — no doutorado os orientadores não apontam um caminho contínuo, mas um caminho cheio de encruzilhadas, nas quais cada escolha leva a um sucesso diferente — por isso, muito obrigado. Lembro, também, do Prof. Alberto, uma espécie de co-orientador "virtual", participando efetivamente de todos os artigos resultantes desta tese.

Adicionalmente ao crescimento acadêmico, neste doutorado, tive um importante crescimento pessoal. Estar no dia a dia do laboratório 210 me proporcionou inúmeras amizades, as quais irei guardar para o resto da minha vida! Nesta sala tínhamos momentos de descontração, discussões políticas e até filosóficas, em resumo, um crescimento pessoal substancial! Algumas pessoas vieram, deixaram sua marca e foram embora (Kindin, Maurício, Maicão, Peru, Marotta, Eder, Muriel, Vilmar, Pedro, Mário, Lauren, Ricardão, Ian, João, Hugo e Leonardo) outras, ficaram ao meu lado até este final de doutorado, tais como, Matias (ninguém vai esquecer aquela final da copa do brasil que você não lembra nada, não é?) e Henrique (quantas ofensas carinhosas!). Outros ainda, conheci aqui e vão construir a nova 210 (Vlad, Mirim, Purinho, Ana, Naldo, Juca, Seninha e Marrone). Ainda há o outro conjunto infinito de colegas dos outros laboratórios do grupo de redes, os quais não vai dar para nominar, de qualquer forma, fica aqui a lembrança. Muitos desses também fizeram parte do grupo Japeca de futebol. Quantos gols, quantas corridas, quantas brigas (não é Peru?). No final das contas, o jogo semanal era um desafogo para o estresse, sendo assim, essencial para este doutorando, obrigado gurizada.

No final do primeiro ano deste doutorado, recebi um desafio do Prof. Lisandro, assumir

a gerência do desenvolvimento dos sistemas da Sociedade Brasileira de Computação - SBC. Apesar desta tarefa não estar diretamente relacionada ao tema desta tese, ela ocupou um espaço de tempo e foi importante para o crescimento profissional. Muitos desenvolvedores estiveram ao meu lado durante o período que fui gerente. Em um primeiro momento éramos Eu, Matheus (um primo que a UFRGS me propíciou conhecer) e o Eder. Depois a equipe foi acrescida por Gustavo e Vinícius. Em sequência, Muriel também se engajou na tarefa, seguido por Pablo, Rafael Viegas, Rafael Ribeiro e Juliano. A todos vocês, muito obrigado pela contribuição.

Um leitor atento desta tese irá perceber que um mesmo conjunto de autores (Muriel, Eder, Arthur e Ricardo Luís) está presente na maioria das publicações resultantes deste doutorado, este grupo de trabalho é a *"papers machine"*. A *papers machine*, nascida na 210, tem uma metodologia própria de cooperação: quando alguém tem uma ideia, esta é sabatinada exaustivamente em reuniões até virar uma proposta de artigo; o autor principal coloca ela no papel, que é revisado por todos os outros. Além disso, é normal que um determinado artigo seja dividido em várias tarefas, executadas por cada membro. A *papers machine* trouxe, adicionalmente à amizade, resultados expressivos com, até a data da entrega desta tese, 10 artigos publicados/aceitos/submetidos. Por isso, agradeço a cada um de vocês, meus amigos!

Em relação à Família, cabe, além dos agradecimentos, um pedido de desculpas. Durante esses quatro anos por diversas vezes limitei as minhas visitas aos meus avós maternos Anita e Silvino, e paternos, Erna e José (*in memoriam*). Também, pouco encontrei tios, tias, primos e primas. Assim, agradeço por vocês entenderem esta ausência. Estendo o mesmo reconhecimento para meus irmãos Pedro Silvino (um irmão, um amigo!) e Paulo Henrique (és um herói!), por entenderem eventual ausência do irmão. Meu mais sincero obrigado também para meus sogros, Albino e Iracema. Além de me concederem o cuidado daquilo que lhes é mais precioso, sua filha Helena, também nos ajudaram muito, cuidando das meninas (Maggie e Punk) sempre que precisamos, e nos recebendo sempre carinhosos. No mesmo sentido, obrigado aos meus cunhados e cunhadas, que sempre nos acolheram muito bem.

Nestes agradecimentos também vale um destaque especial, aos meus pais, Prof. Mestre Paulo e Profa. Doutora Elisete. Não se surpreendam por eu utilizar os teus títulos ao nomeá-los, eles são muito importantes! Pais, além de serem os primeiros educadores, devem ser inspiração, modelo, para seus filhos. Vocês cumpriram esse papel com excelência, muito obrigado pelos ensinamentos e pela vida honrada que vocês me propiciaram e inspiraram!

Por último e mais importante, destaco a participação da minha companheira de lutas e vitórias, a minha amada esposa Helena. Só nós sabemos o que vivemos. O desenvolvimento do doutorado nos consumiu muito, exigiu força, companheirismo, paciência, amor, paixão e todas as outras qualidades que um casal tem para oferecer mutuamente. Não foi só pelas exigências da pós-graduação em si — que convenhamos, não é para fracos (lidar com a rejeição é só um exemplo daquilo que suga o nosso âmago) — haviam as viagens, os desafios financeiros, as noites sem dormir, os pagodes dos vizinhos e tantas outras adversidades. Mas mesmo assim, você sempre esteve lá! Muito obrigado por tudo, te amo muito!

## ABSTRACT

Network functions perform an important role in the communication between clients and end-services in computer networks. These functions, traditionally provided by hardware-based appliances, are employed to achieve various objectives, such as security, addressing, routing, and load balancing. The virtualization trend that drove cloud computing has also been applied to network functions, resulting in the Network Functions Virtualization (NFV) concept. NFV implements hardware-based network functions (a.k.a middleboxes) as Virtual Network Functions (VNFs); with this paradigm shift, network providers benefit from: capital and operational expenditure reduction, acceleration of the time-to-market of novel solutions, and resource scaling. One of the main features of NFV is that VNFs can be chained and provisioned on demand, bringing elasticity and dynamicity to the network. An implication of the interdependency between VNFs is that resulting service chains may not work as expected, and because of that, it is crucial to determine which VNFs are having a negative impact on the service quality. Usually, network operators detect bottlenecks and quantify performance degradations by monitoring both virtual and physical resources, and through analytical modeling of the various components involved in network communication. However, NFV imposes additional characteristics that hinder the adoption of these approaches: the heterogeneity of environments with all sorts of VNF purposes (*e.g.*, provide security, address translation, and performance optimization), and the unusual functioning of specific VNFs that rely on non-blocking I/O implementations.
In this thesis, a model to quantify the `guiltiness` of a VNF on being a bottleneck in a service chain is proposed. This model consists of a novel application of the *Utilization Law* from queuing networks theory, combined through a weighted sum of relevant metrics. These metrics were found in the literature, refined through numeric assessments on representative scenarios. In addition, an adaptive algorithm based on linear regression and neural networks is introduced to adjust the model parameters according to the environment particularities, such as the type and number of VNFs in the service. Experimental evaluations confirm the ability of the model to (*i*) detect bottlenecks, and (*ii*) quantify performance degradations. When capacity restrictions were applied to different types of VNFs, `guiltiness` metric was able to detect the root cause of degradations, being able to characterize 96.15% of the performance issues in a scenario with controlled injections. Furthermore, studies are carried out to establish the appropriate method for monitoring NFV environments. To this end, the Distributed Result-Aware Monitoring (DReAM) approach is proposed: a scalable monitor that can achieve near real-time bottleneck detection.



**Keywords:** Network Functions Virtualization. Bottlenecks Identification. Performance Analysis.

**Monitorando e Identificando Gargalos em Cadeias de Serviço de Funções de Rede Virtualizadas**

## RESUMO

Em redes de computadores as funções de rede têm um papel significativo na comunicação entre clientes e serviços finais, sendo responsáveis por realizar diferentes tarefas, tais como, segurança, endereçamento, roteamento, *caching* e balanceamento de carga. Seguindo os passos que impulsionaram a consolidação da computação em nuvem, uma tendência atual tem sido virtualizar as funções de rede, estabelecendo um novo conceito: *Network Functions Virtualization* (NFV). Conceito este que tem se firmado como paradigma base das arquiteturas de redes de computadores do futuro. Em resumo, NFV permite, através de uma camada de virtualização, que funções de rede que executam em hardware dedicados tornem-se elementos de software independentes de hardware subjacentes, sendo então, nominadas como *Virtual Netwok Functions (VNFs)*. Dentre os benefícios de NFV pode-se citar: redução de custos operacionais, agilidade para o estabelecimento de novas soluções e elasticidade de recursos. Além disto, uma das principais caraterísticas de NFV é a possibilidade do encadeamento de VNFs sob demanda, o que torna a composição de serviços dinâmica. Neste contexto, provedores de serviço utilizam grafos de encaminhamento para construir cadeias de VNFs para compor serviços de rede que atendam as demandas particulares de seus clientes. Como consequência, o desempenho de uma VNFs pode impactar as outras que compõem a cadeia, e assim, é crucial aos operadores de rede poder identificar quando, e o quanto, uma determinada VNF impacta negativamente a qualidade do serviço. Para lidar com esse problema, operadores de rede geralmente monitoram o consumo dos recursos e aplicam modelos analíticos, para, respectivamente, detectar gargalos e quantificar as degradações de desempenho. Entretanto, algumas características intrínsecas à NFV limitam a utilização dessas abordagens: os ambientes de NFV são heterogêneos, ou seja, as VNFs atendem a diversos propósitos (*e.g.*, segurança, endereçamento e melhoramento de desempenho); e, algumas implementações de VNFs incorrem em chamadas de E/S não bloqueantes para melhorar o seu desempenho.

Para lidar com estas limitações, nesta tese é proposto um modelo para quantificar a propensão de uma VNF ser um gargalo da cadeia de serviços. A métrica resultante desta proposta é denominada `guiltiness` (culpabilidade em português), e consiste em uma soma ponderada de um conjunto de métricas relevantes. Este conjunto foi refinado, através de análises experimentais, do conjunto total de métricas disponível na literatura atual de NFV. O uso de pesos na métrica permite que a mesma seja aplicada mesmo em cenários heterogêneos, neste caso, um algoritmo de adaptação baseado em regressões não-lineares e redes neurais calcula os valores do pesos de acordo com as particularidades de cada cenário. Para comprovar a aplicabilidade do modelo na identificação de gargalos e na estimativa de degradação de desempenho, foram

conduzidas avaliações experimentais em cenários representativos da literatura. Os resultados destas avaliações mostram que a métrica de `guiltiness` serve ao seu propósito, uma vez que caracterizou corretamente as degradações de vazão quando gargalos eram gerados por limitação de recursos, e identificou 96.15% dos problemas de desempenho em um cenário com sobrecargas artificialmente injetadas. Adicionalmente à identificação de gargalos e estimativa de desempenho, estudos foram realizados para estabelecer o método apropriado de monitoração de ambientes NFV. Para este propósito, esta tese propõe o uso de uma abordagem de monitoramento distribuída e baseada no resultado (*DReAM – Distributed Result-Aware Monitor*), para tanto, agentes de monitoração computam um diagnóstico para identificar o estado das VNFs. Como principal benefício, esta abordagem permite identificar, quase imediatamente, quando uma VNF se torna um gargalo para o serviço de rede.
**Palavras-chave:** Virtualização de Funções de Rede, Identificação de Gargalos, Análise de Desempenho.

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| ACL | Access Control List |
| ADSL | Assymetrical Digital Subscriber Line |
| API | Application Programming Interface |
| ASP | Application Service Provider |
| ANOVA | Analysis of Variance |
| BBU | Base Band Unit |
| BRAS | Broadband Remote Access Server |
| BS | Base Station |
| CaaS | Crypto as a Service |
| CAPEX | Capital Expenditure |
| CDN | Content Delivery Networks |
| COTS | Commercial Off-The-Shelf |
| CPU | Central Processing Unit |
| cps | Calls per second |
| CRAN | Cloud Radio Access Networks |
| CSCF | Call Session Control Function |
| CTPN | Colored Time Petri Network |
| DFC | Distributed Flow Switch Controller |
| DNS | Domain Name System |
| DNE-RTC | Dynamic Network Enabled Real-Time Control |
| DPI | Deep Packet Inspection |
| DReAM | Distributed Results-Aware Monitor |
| DSL | Digital Subscriber Line |
| EPC | Evolved Packet Core |
| E/S | Entrada/Saída |

| | |
|---|---|
| ESP | End-Service Provider |
| ETSI | European Telecommunications Standards Institute |
| FG | Forwarding Graph |
| FPR | False Positive Rate |
| FW | Firewall |
| GB | Gigabytes |
| Gbps | Gigabits per second |
| GHz | Gigahertz |
| HSS | Home Subscriber Server |
| HDD | Hard Disk Drive |
| HTTP | Hypertext Transfer Protocol |
| I/O | Input/Output |
| I-CSCF | Interrogating-CSCF |
| IDS | Intrusion Detection System |
| IETF | Internet Engineering Task Force |
| IMS | IP Multimedia Subsystem |
| IPS | Intrusion Prevention System |
| IP | Internet Protocol |
| IoT | Internet of Things |
| IPSec | Internet Protocol Security |
| IRTF | Internet Research Task Force |
| ISC | IP multimedia Service Control |
| ISP | Internet Service Provider |
| KB | Kilobyte |
| Kbps | Kilobits per second |
| KPI | Key Performance Indicators |
| LAN | Local Area Network |
| L2 | Level 2 |
| L3 | Level 3 |

| | |
|---|---|
| LB | Load-Balancer |
| LLC | Last Level Cache |
| LTE | Long-Term Evolution |
| MANO | Management & Orchestration |
| MAI | Mobile App Identification |
| MB | Megabytes |
| MbD | Management by Delegation |
| Mbps | Megabits per second |
| MiMP | Man-in-the-middle Proxy |
| MGW | Media Gateway |
| MLM | Mid-level Managers |
| MME | Mobility Management Entity |
| NAT | Network Address Translation |
| NDN | Named Data Networking |
| NF | Network Function |
| NFD | NDN Forwarding Daemon |
| NFV | Network Functions Virtualization |
| NFVI | NFV Infrastructure |
| NFVIaaS | NFV Infrastructure as a Service |
| NFVO | NFV Orchestrator |
| NFVRG | NFV Research Group |
| NFVSP | NFV Service Provider |
| NIC | Network Interface Card |
| NS | Network Service |
| NSd | Network Service Descriptor |
| NVP | Network Virtualization Proxy |
| OPEX | Operational Expenditure |
| OS | Operating System |
| PC | Personal Computer |

| | |
|---|---|
| PCA | Principal Component Analysis |
| PCE | Path Computation Element |
| P-CSCF | Proxy-CSCF |
| PCRF | Policy and Charging Roles Function |
| PGW | Packet Data Network Gateway |
| PKI | Public Key Infrastructure |
| PMIPv6 | Proxy Mobile IPv6 |
| PNF | Physical Network Functions |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| $R^2$ | R-squared |
| RAN | Radio Access Networks |
| REST | Representational State Transfer |
| RFC | Request for Comments |
| RQ-1 | Research Question 1 |
| RQ-2 | Research Question 2 |
| RQ-3 | Research Question 3 |
| RTT | Round-trip delay time |
| RX | Receive |
| S-CSCF | Serving-CSCF |
| SDK | Software Development Kit |
| SDN | Software-Defined Networking |
| SDRAN | Software-Defined Radio Access Networks |
| SEP | Symantec Endpoint Protection |
| SecaaS | Security as a Service |
| SFC | Service Function Chain |
| SGW | Serving Gateway |
| SIP | Session Initiation Protocol |
| SLA | Service-Level Agreement |

| | |
|---|---|
| SP | Service Provider |
| SR-IOV | Single Root Input/Output Virtualization |
| SRV | Service Record |
| STB | Set-Top Boxes |
| TCP | Transmission Control Protocol |
| TLM | Top-Level Managers |
| TPR | True Positive Rate |
| UCON | Usage Control |
| UDP | User Datagram Protocol |
| VDSL2 | Very-High-Bit-Rate Digital Subscriber Line 2 |
| VLAN | Virtual Local Area Network |
| VIM | Virtualized Infrastructure Manager |
| VN | Virtual Network |
| VNF | Virtual Network Function |
| VNFd | Virtual Network Function descriptor |
| VNF FG | Virtual Network Function Forwarding Graph |
| VNFM | Virtual Network Function Manager |
| VM | Virtual Machine |
| vNIC | virtual Network Interface Card |
| VOC | Video Optimization Controller |
| VoIP | Voice over IP |
| WAF | Web Application Firewall |
| WAN | Wide Area Network |
| WGW | Wireless Gateway |
| WOC | WAN Optimization Controller |
| XCAP | XML Configuration Access Protocol |
| XDMS | XML Document Management System |

## LIST OF FIGURES

## LIST OF TABLES

## CONTENTS

# 1 INTRODUCTION

Network functions perform an important role in the communication between clients and end-services in computer networks. These functions are employed to achieve numerous objectives, such as providing security, addressing, routing, caching, and load balancing (NANDUGUDI et al., 2016). In traditional networks, network functions run in dedicated hardware middleboxes, which restricts the development of novel solutions to specific vendors, impacting negatively on the CApital EXpenditure (CAPEX) and OPerational EXpendinture (OPEX) of companies (NGUYEN; DO; KIM, 2016). Recently, the virtualization trend that drove cloud computing has also been applied to network functions, allowing Virtual Network Functions (VNF) to run in Commercial-Of-The-Shelf (COTS) servers, decoupled from the underlying hardware (MIJUMBI et al., 2015a). In this context, both industry and academia have been carrying out efforts to develop and standardize Network Functions Virtualization (NFV). The European Telecommunications Standards Institute (ETSI) defined an Industry Specification Group (ISG) to study, standardize, and develop NFV; in another movement, the Internet Research Task Force (IRTF) created the NFV Research Group (NFVRG) to deal with NFV-related research problems.

One of the use cases proposed by the ETSI NFV ISG refers to the ability of VNFs being chained and provisioned on demand, bringing elasticity and dynamicity to the network (ETSI NFV ISG, 2017). In particular, Service Providers (SPs) take advantage of forwarding graphs (FGs) and Software-Defined Networks (SDN) technologies to specify service chains, which consist of a pre-established order of VNFs that network packets must traverse, and provide customized network services (NSes) to meet individual client demands (MUHAMMAD et al., 2016). For instance, an NFV Service Provider (NFVSP) may offer a forwarding graph portfolio, including chains for network performance improvement, security inspections, and voice calls registering. In such scenario, each VNF has a certain impact on the overall network service performance, depending on several factors (*e.g.*, implementation, provisioned resources, and chaining order) (HAN et al., 2015). Hence, network operators should be able to quantify these impacts to spot bottlenecks, identifying the VNFs that are degrading the performance of the chain. Also, even if the network is not experiencing performance degradations, the knowledge of how much each VNF affects network performance can help operators in planning decisions, since providing more resources to the VNF with the most impact on performance will likely enhance Quality-of-Service (QoS) and Quality-of-Experience (QoE).

Usually, operators rely on monitoring physical and virtual resources to point out VNFs that are causing performance issues. This detection method consists in measuring metrics' values at the Operating System (OS) level (*e.g.*, CPU usage, memory usage, packet loss) and comparing them to pre-established thresholds (PFITSCHER; PILLON; OBELHEIRO, 2014; PFITSCHER; PILLON; OBELHEIRO, 2013). When a given VNF measurement surpasses such thresholds, the monitoring system triggers an event to inform the network operator that

the VNF is hindering the performance of the service chain. However, such threshold-based approach does not help network operators accurately relate a metric to the performance of an end-service that relies on the service chain. Another aspect that limits the usage of OS level monitoring is the heterogeneity of NFV environments, which are characterized by the diversity of network function types. As a result, a chain for a given service type can be more influenced by a particular resource than others. For example, while the performance of VNFs running in IP Multimedia Subsystems (IMS) is more sensitive to network metrics (*e.g.*, latency, packet loss, and throughput), the performance of security-related VNFs (*e.g.*, firewall and Deep Packet Inspection (DPI)) is more affected by CPU metrics.

One approach commonly applied to quantify performance and detect bottlenecks is to represent the various components involved in communication through analytical models (DENNING; BUZEN, 1978; HARCHOL-BALTER, 2013; GUNTHER, 2007). For instance, in traditional queueing networks, each node is modeled as a queue and a processing component: the queue stores the requests that are processed and forwarded to the next node. In such approach, historical resource usage information is applied in mathematical formulations to predict the residence time in each node; then, to quantify performance impacts, one must compare the predicted residence times. However, since the VNFs of a service chain intermediate the communication between clients and end-services, performance degradations must be detected in near real-time (YI et al., 2018). Such requirement limits the use of traditional mathematical models, because of the offline nature of analytical performance analysis and the need of historical information. In the context of NFV, both threshold monitoring (PFITSCHER et al., 2016; GEMBER et al., 2013; CAO et al., 2015) and analytical modeling (GALLARDO; BAYNAT; BEGIN, 2016; SUKSOMBOON et al., 2016; KOIZUMI; FUJIWAKA, 2015) have been exploited in recent literature to detect bottlenecks. However, such efforts require a low-level instrumentation of the network, or even a pre-monitoring calibration step, to accurately identify bottlenecks. Such requirements significantly hinder the feasibility of bottleneck detection solutions, as they are most commonly impossible to meet in real-world scenarios, with multiple stakeholders and restricted access/control over networks.

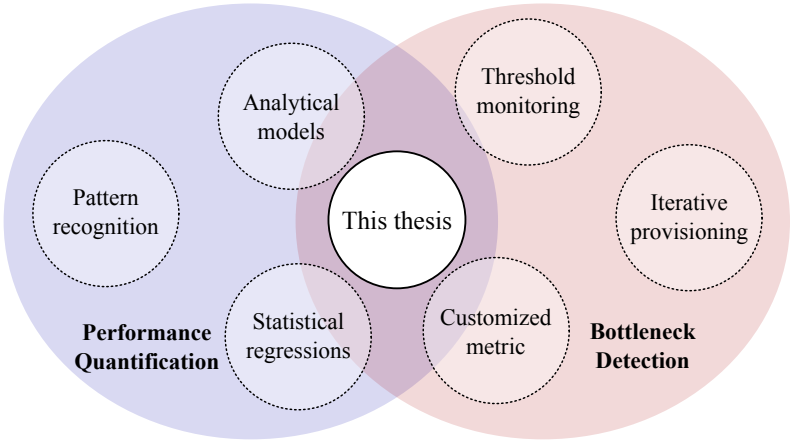
In this Thesis, we address the problem of measuring how much each VNF impacts in NFV service function chains by proposing a metric able to quantify performance degradations. Our first step towards this objective was to assess the representativeness of state-of-the-art metrics in distinct scenarios. Through empirical observations, we reduce the set of monitoring metrics to a subset containing only metrics representative of performance in multiple scenarios (*i.e.*, CPU usage, memory usage, Network Interface Card (NIC) usage, I/O usage, and NIC queueing). Then, by introducing the activity of resources in concepts from queueing network modeling, we combine the subset of metrics in a weighted-sum and define the `guiltiness` of each VNF on degrading the service chain performance. Such metric consists of an *online* estimate that allows operators to identify the health of running VNFs in a given service chain. Also, to increase the accuracy of estimates, we propose a learning algorithm that combines non-linear

regressions and neural networks to adjust the model according to the environment.

## 1.1 Research scope

This Thesis addresses the problem of bottleneck identification in NFV service chains. Hence, we aim to provide the rules of thumb for both the quantification of service chain performance degradation and the identification of bottleneck VNFs. Figure 1.1 illustrates the expected position of this Thesis regarding the state-of-the-art. For the sake of readability, this Section briefly discusses each work according to their application and employed method. In turn, an in-depth analysis can be found in Section 2.5.

Figure 1.1: Thesis positioning




Source: the Author

Bottleneck detection can be highlighted by three approaches: first, *threshold monitoring* approaches (PFITSCHER et al., 2016; GEMBER et al., 2013; CAO et al., 2015) are commonly used to detect resource contention-related performance degradations. As mentioned above, this method consists of monitoring a set of metrics and notifying operators when measurements cross a predefined level. Second, other works (WU; HE; AKELLA, 2015; NAIK; SHAW; VUTUKURU, 2016) propose *customized metrics* for particular scenarios. Such approach consists of either combining multiple metrics or leveraging deep instrumentation to collect non-traditional performance indicators (*e.g.*, CPU backlog, NIC backlog, and page faults). Third, through *iterative resource provisioning*, as used by Gember et al. (2013) and Cao et al. (2015), operators can scale up and down provisioned resources of each VNF and verify whether such actions improve or decrease the performance of the NFV service chain.

The performance quantification problem has been well stressed in the computer networks literature. In terms of its application to NFV, the three major approaches consist of: developing *analytical models*, performing *statistical regressions*, and applying *pattern recognition* techniques. In analytical models (SUKSOMBOON et al., 2016; KOIZUMI; FUJIWAKA, 2015;

GALLARDO; BAYNAT; BEGIN, 2016) the performance of VNFs is mathematically predicted through a set of equations. For example, in queueing theory, each resource involved in the VNF functioning is modeled as a queue, allowing to compute the time each request spends in each resource. However, these models depend on historical observation of monitoring data to achieve accurate results. A quite similar approach is to rely on statistical regressions to compute weights of mathematical equations. In fact, statistical regressions can be considered a step to predict performance through analytical models. Finally, pattern recognition approaches (SAUVANAUD et al., 2016) rely on monitoring data and machine learning algorithms to characterize VNFs performance in multiple levels. The approach consists in correlating monitoring metrics of VNFs to a specific performance level; thus, when a VNF behaves similarly to a previous pattern, it is possible to infer that the performance will be also similar.

Given this range of methods and its twofold application, as shown in Figure 1.1, this Thesis contributes to the area with a model able to: (*i*) accurately quantify the relevance of each VNF in the service chain performance, (*ii*) point out bottleneck VNFs, and (*iii*) self-adjust to environment particularities. To this end, we *analytically model* a *customized metric*, use *statistical regressions* to adjust the model according to scenarios, and leverage a *threshold monitoring* approach to detect bottleneck VNFs of NFV service chains. In the following section, we discuss the limitations of the above approaches and define our research problem as well as the research questions.

## 1.2 Problem statement and research questions

The state-of-the-art, which is thoroughly discussed in Section 2.5, presents a broad set of metrics used for NFV service chains performance diagnosis, increasing the complexity for operators to analyze various NFV scenarios. A common characteristic of metrics proposed so far in the literature is that they are tailored to meet specific scenarios particularities. For instance, NIC queueing and packet loss analysis, leveraged by Wu, He and Akella (2015), mainly applies to network-caused issues. Also, resource usage metrics (*e.g.*, CPU, I/O, and memory) may have different levels of impact on performance according to the function that the VNF performs. Such metrics' imprecisions demand NFV operators to have an in-depth knowledge of their environments, and thus, they must perform several tests in each of their managed setups to find out which metrics are more representative for their scenarios. However, testing each environment is prohibitive, given the heterogeneity of NFV, which is usually composed of multi-purpose VNFs and also can delay the diagnosis process.

In addition to the plethora of metrics, the state-of-the-art also presents multiple approaches for performance quantification and bottleneck detection. A set of major concerns arise when solving these problems simultaneously: threshold monitoring approaches are not able to quantify performance degradations and they require a calibration step to accurately diagnose bottlenecks; customized metrics depend on low-level instrumentation of operating system queues, or

need access to network devices to perform mirroring; iterative provisioning can delay the bottleneck diagnosis process and fail to quantify degradations; proposed analytical models require a deep knowledge of each type of VNF and their internal communications; and pattern recognition and statistical regressions need a large set of results before providing precise estimates of performance. Considering these limitations on diagnosing NFV performance we define the following research problem.

**Problem definition:** *The problem addressed in this Thesis is the lack of an approach that can, at the same time, both detect bottlenecks in NFV service chains and quantify the likelihood of individual VNFs affecting communication performance in general-purpose NFV scenarios.*

We formulate three research questions to pave the road towards solving the problem above. The first research question concerns the use of traditional metrics and their ability to quantify the performance of NFV service chains. As we previously discussed, the two approaches that are most commonly used for diagnosis of the VNFs performance problems (threshold-based monitoring and analytical modeling) rely on historical observations of traditional metrics. These metrics include, but are not limited to, CPU usage, memory usage, network throughput, packet losses, and latency between nodes. However, given the particular characteristics of each VNF in the chain, a deeper study is required to point out the relevance of such metrics on portraying the overall chain performance. Among such characteristics, we highlight two of them: VNFs that exhibit 100% CPU usage because of the non-blocking I/O implementation, and the variance in sensitivity of each performance metric due to the heterogeneity of VNFs purposes. In view of this, we formulate the first Research Question (RQ-1):

**RQ-1:** *Given that the diagnosis of NFV performance has to take into account VNFs' particular characteristics, what is the set of representative metrics able to quantify the performance impacts imposed by each VNF in a service chain?*

By answering RQ-1, we define the set of metrics that can represent the variations on the end-services' performance in general NFV scenarios. Such answer shows that one may have to rely on multiple metrics for service chain performance quantification, since the metrics representativeness varies according to the type of service being monitored. To avoid such ambiguity, diagnosis based on these metrics must combine their measurements in a way that consider their specific relevance in each particular scenario. However, similarly to the metrics case, the state-of-the-art also present numerous methods to this end, including machine learning (PASQUINI; STADLER, 2017), weighted sums (KOIZUMI; FUJIWAKA, 2015), queueing networks models (SUKSOMBOON et al., 2016), and threshold-based (CAO et al., 2015). The problem of deciding on what method to use has brought us to the second Research Question (RQ-2):

**RQ-2:** *Having found a set of performance representative metrics, how can they be combined in a model to accurately quantify performance degradations caused by individual VNFs in general NFV scenarios?*

By answering RQ-2, we establish a model that produces an estimate of the performance degradation caused by each VNF in a service chain. Such model consists of a novel application of well-established queueing network laws, which combines both resource activity measurements and weights to gauge the importance of each metric in the set. While the active time provides an *online* estimate of metrics' importance, weights must be fine-tuned according to the particularities of the service chain being evaluated. Several machine learning algorithms may apply to this adaptation problem, including neural networks, pattern recognition methods (*e.g.*, Principal Component Analysis (PCA)), regressions, and statistical methods (*e.g.*, Analysis of Variance (ANOVA)). The decision of which one to select and how to define weights for resources lead us to the last Research Question (RQ-3):

**RQ-3:** *Considering that weights of the model should be adjusted to scenarios' particular characteristics, how to adjust the diagnostic model according to varying setups in scenarios?*

By answering RQ-3, we introduce a suitable method to maintain the performance diagnostic accurate in general scenarios. Two main aspects were taken into account to provide such a solution: as VNFs may exhibit anomalies in resource consumption, the adjustment must be flexible to abstract outliers, and; because VNFs intermediate communication between clients and end-services, the adjustment process must run *online*. We introduce a hybrid learning mechanism that combines statistical regressions and neural networks, which allows weights adjustment even when scenarios vary significantly. The mechanism also uses monitoring information from VNFs to perform model adjustments online. Experimental assessments show the benefits of the proposed hybrid mechanism regarding precision improvements when compared to default weights.

## 1.3 Main Contributions

During the process of solving the research problem above and answering the related research questions, many advancements on the state-of-the-art are expected. This Thesis presents the following contributions:

1. An analysis of the current picture of evaluation scenarios in NFV, detailing monitoring metrics, VNF types, service chain sizes, and workloads used by researchers to endorse their proposals in the NFV context;
2. An experimental characterization of NFV performance in distinct scenarios, providing an extensive analysis of most commonly used monitoring metrics. With such characterization, we present a detailed evaluation of how much each monitoring metric is able to accurately identify bottlenecks and quantify performance degradation in NFV environments;
3. A novel, practical, mathematical model that is able to accurately identify bottlenecks

and quantify performance degradation for individual VNFs in a virtualized service chain. With such model, operators are able to pinpoint which VNFs have higher impact on end-service performance and hence more precisely act to improve user experience by either optimizing resource provisioning or preventing errors with informed planning decisions.

## 1.4 Thesis Organization

The remainder of this Thesis is organized as follows. In Chapter 2, we review the central NFV concepts and the studies related to this Thesis. First, we provide an overview of NFV and its evolution, as well as describe the leading use cases of NFV application. Afterwards, we detail the methods and metrics used in the state-of-the-art to analyze VNFs performance. We split the discussion of works into two categories. In the first one, we discuss the works that aim to propose methods for bottleneck detection. In the second one, we detail the methods and metrics used for NFV service chains performance quantification.

In Chapter 3, we address the first research question. For the sake of generality, we first review the literature to identify the most common NFV service chain applications, establishing straightforward base-scenarios to apply our evaluations. After that, we define the set of reference metrics used in performance evaluation, as well as the bottleneck identification methods published in related work. Next, aiming to assess how each metric behaves in the base-scenarios, we perform experimental evaluations to monitor metrics' behavior against reference workloads from the literature. Finally, we analyze the results and define the set of metrics that can represent VNFs performance in general scenarios.

In Chapter 4, we continue our research effort by addressing the second and the third research questions. Therefore, we establish a model to represent VNFs individual impacts on performance by combining the relevant metrics. This modeling process consists of an analytical evaluation of service chain performance components. Next, we appraise methods to adjust the variable part of the model according to environment particularities. First, we research about learning algorithms techniques suitable to the adjustment objective. Then, we introduce a hybrid learning mechanism that comprises regressions and neural networks to learn model weights *online*. Finally, through experimental evaluations, we depict the benefits of such a learning approach regarding the accuracy of detected bottlenecks.

In Chapter 5, we present the design of a solution to compute the performance model and monitor the VNFs state. To accomplish the near real-time state change detection requirement of NFV, we rely on a result-aware distributed monitoring approach that runs diagnostic models inside VNFs to provide state results, and also collects neighborhood information to point out bottlenecks. We performed experimental evaluations to assess the solution scalability and its ability to detect performance issues.

In Chapter 6, we discuss the final remarks of the Thesis and present future works. The research questions are revisited to discuss the answers we provide and to point out the remaining

gaps subject of future investigations. Also, we make explicit the limitations of our proposal, delimiting the suitable cases of application.

# 2 BACKGROUND AND STATE-OF-THE-ART

Aiming to provide a clear understanding of the challenges inherent to performance quantification and bottleneck detection in NFV, in this Chapter, we review base concepts and discuss the state-of-the-art. To that end, we present the background knowledge regarding the NFV concept, the leading use cases for the technology, the performance implications of its application, and the state-of-the-art of NFV performance analysis.

## 2.1 Network Function Virtualization

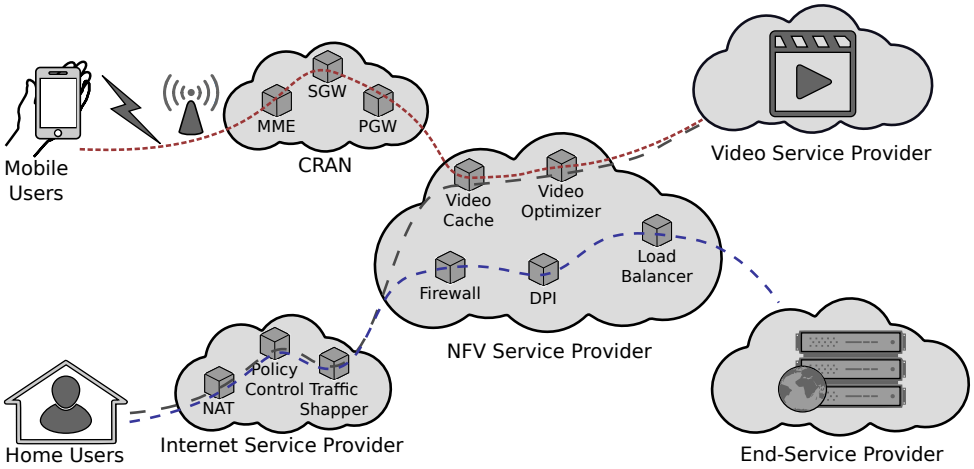
Network services in the telecommunication industry have traditionally relied on network operators deploying each network function as a proprietary hardware device, commonly referred to as middleboxes. However, using hardware-based network functions to provision network services led the industry to face an ever growing need of physical space, energy, and specialized knowledge to integrate and operate such devices. In addition, as technology advances, investments on new proprietary devices are constantly needed to maintain service quality, resulting on a slow-paced innovation cycle and longer Time-to-Market.

The NFV concept was proposed by the European Telecommunications Standards Institute (ETSI) (ETSI NFV ISG, 2012) aiming to address the challenges derived from using physical appliances in network service provisioning. NFV aims to migrate middleboxes from dedicated hardware to software running on Virtual Machines (VMs), referred to as VNFs, which can be deployed on Commercial-of-the-Shelf (COTS) servers. By decoupling network functions from hardware, network operators are able to dynamically create, deploy, migrate and remove functions in the network. Further, enabling NFV in networks reduces operational and financial costs, since less investments are needed in hardware devices. In general, NFV achieves increased scalability, flexibility and availability, as well as avoids the ossification of the network.

Figure 2.1 illustrates three use cases of NFV application in the current network scenario (ETSI NFV ISG, 2013). In the first case, a video service provider offers video-on-demand for its users. With the aim of improving the Quality of Experience (QoE) of its clients, it hires two VNFs from an NFV Service Provider (NFVSP): a video cache and a video optimizer. To achieve QoE improvement, a common strategy of NFVSPs is to provision instances of VNFs closer to end users, which reduces the access latency. In the second case, with the aim of reducing costs, End-Service Providers (ESPs), such as e-commerces and web-servers, acquire VNFs from the NFVSP that are not directly related to ESPs' end-service (e.g, firewalls, DPIs and load balancers). In the third case, Internet Service Providers (ISPs) and Cloud Radio Access Networks (CRAN) also benefit from NFV. The former uses NFV to reduce the costs of set-top boxes (STB) by replacing them with VNF instances, according to the clients demands. The latter uses virtualization on the full stack of network protocols (MAROTTA et al., 2015; ROST et al., 2014), in which a Mobility Management Entity (MME) is responsible for receiving signals and

choosing the appropriate Serving Gateway (SGW), through which the traffic passes through. Then, the traffic is forwarded to the Packet Gateway (PGW) that transmits data packets to the external network.

Figure 2.1: NFV use case examples

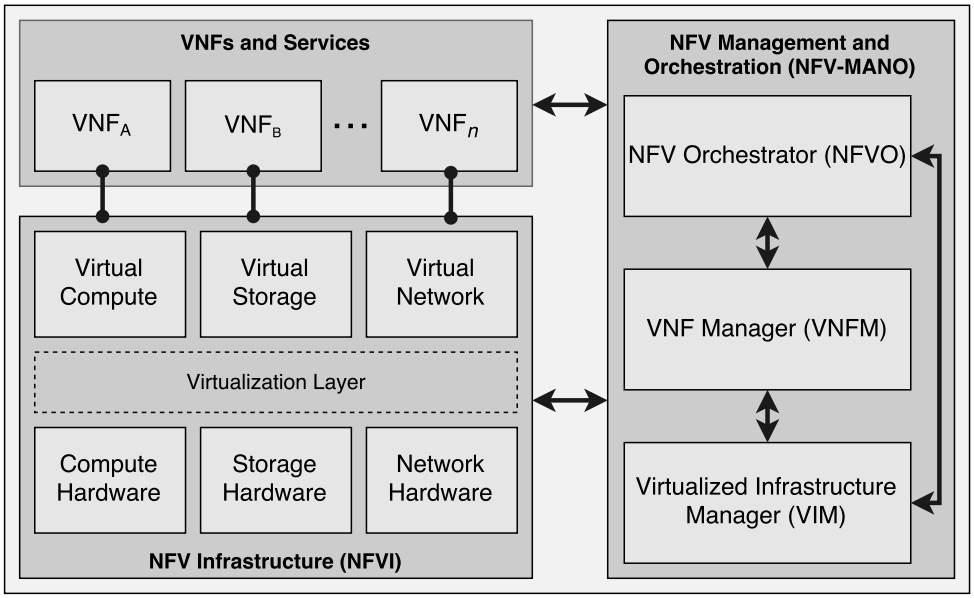



Source: based on (ETSI NFV ISG, 2013)

The intrinsic advantages and innovation granted by the NFV concept also introduce a multitude of challenges and issues regarding management and infrastructure. Since network functions are virtualized in NFV environments, they need to be deployed on an agnostic NFV Infrastructure (NFVI), with carrier-grade servers, data centers and network, alongside virtualization enablers, such as hypervisors and VMs to host VNFs. As these entities are exposed by the advent of NFV, the new set of relationships between those entities requires new management solutions. To address these challenges, the ETSI proposed the Management and Orchestration architectural framework (MANO) (ETSI NFV ISG, 2014), which aims to manage the NFVI and orchestrate the resource allocation for VNFs and network services.

The overall architecture proposed by the ETSI can be seen in Figure 2.2. The NFVI corresponds to the underlying compute, storage and network infrastructure of the architecture. The VNFs, and services provided by FGs, are deployed on top of the NFVI. Finally, both the NFVI resource provisioning and orchestration of VNFs and FGs are performed by the NFV-MANO, composed of three distinct functional blocks: the NFV Orchestrator (NFVO), responsible for allocating resources from the NFVI, and for the life-cycle management of network services; the VNF Manager (VNFM), responsible for the creation, migration and deletion of VNF instances; and the Virtualized Infrastructure Manager (VIM), which is in charge of controlling and provisioning NFVI resources to be allocated by the NFVO.

Figure 2.2: NFV conceptual architecture




Source: the Author

## 2.2 NFV Use Cases

Aiming to provide a better understanding of the advantages of NFV and its applications, the ETSI published a document (ETSI NFV ISG, 2013) introducing a number of NFV Use Cases and discussing how existing network scenarios may take advantage of the technology. Such document was updated in 2017 according to the advances on NFV technology (ETSI NFV ISG, 2017). Below, we describe each of the use-cases presented by the ETSI.

**#1 NFVI as a Service (NFVIaaS)**: in cloud computing, a traditional service offer is the Infrastructure as a Service (IaaS), which is the capability to provide consumers the processing, storage and fundamental resources of the cloud. In this scenario, the consumer is able to choose what to use the infrastructure for, without having complete control over it. Based on this service model, the NFVI as a Service offer was proposed, considered as the capability of providing an environment in which VNFs can execute. In this case, the NFVI provides compute, storage and network to consumers to deploy their VNFs and network services, acquiring resources on-demand, but without having control over the NFVI, similarly to an IaaS cloud service.

**#2 VNF Forwarding Graphs (VNF FG)**: a Forwarding Graph (FG) defines a logical sequence of network functions for packets to traverse, providing a network service. In turn, a VNF FG provides a sequence of virtual appliances (*i.e.*, VNFs) which packets must traverse. Aiming to achieve the goals of NFV, service providers must be able to instantiate network services over the NFVI in an abstract level. Such abstract definitions may

include the identification of topology of the network, the VNFs involved and their particular interconnections. Having these abstract definitions, service providers would then be able to deploy the network service over the NFVI, hence taking advantage of several benefits of NFV, such as flexibility and efficiency.

**#3 Virtualization of Mobile Core Network and IMS**: mobile networks are composed of a large variety of proprietary hardware-based network functions. NFV aims to reduce the complexity and operational issues from the network by virtualizing and consolidating the network function into COTS servers. By virtualizing the Evolved Packet Core (EPC), a center network architecture of mobile networks, composed of several different network functions; and the IP Multimedia Subsytems (IMS), a session control architecture to support the provisioning of multimedia system over the EPC, it is possible to achieve greater availability, optimization of the networks utilization, elasticity and topology reconfiguration.

**#4 Virtualization of Mobile Base Station**: this Use Case consists of applying NFV on the Mobile Base Station, virtualizing Radio Access Networks (RAN) and deploying them on standard servers, storage and switches. Virtualizing at least a part of the RANs in the Mobile base station is expected to provide lower footprints and energy consumption by balancing the traffic load and dynamically allocation resources when necessary. In addition, this approach would yield easier management and operation, as well as faster Time-to-Market.

**#5 Virtualization of the Home Environment**: in modern home environments, several services may be contracted from network providers (*e.g.*, cable television and Internet access), which typically requires a dedicated hardware device for each service contracted. In this scenario, NFV may bring several benefits by consolidating all those dedicated network functions into a single equipment with access to the VNFs provided by the network provider, reducing both operational and financial costs, as well as facilitating upgrades to the network service.

**#6 Virtualization of Content Delivery Networks**: in recent years, delivery of content, especially video, has become one of the major challenges for operator networks, due to the massive growth of traffic to be delivery to end-users with the advent of streaming companies such as Netflix. In this context, virtualizing Content Delivery Networks (CDNs) and integrating them into operator networks might improve QoS and the scalability, via transparent caching.

**#7 Fixed Access Network Functions Virtualization**: access network is often stated as the main cause of costs and bottlenecks, due to limited bit rates in current access technologies (*i.e.*, DSL and ADSL2). The current trend, however, is to deploy equipment based on VSDL2, using fiber to achieve higher bit rates. VSDL2 require systems to be deployed on remote nodes located on streets or in multiple-occupancy buildings. In this scenario,

having low costs, minimal power consumption and lower complexity in the remote nodes, which can all be achieved by benefiting from NFV technology.

**#8 Crypto as a Service (CaaS)**: with the advent of NFV, critical security network functions, such as firewalls and IPSs, are being moved to cloud-like environments. As a result, these VNFs are faced with performance, scaling and key management issues. The CaaS sheds light in the need for high-performance cryptographic key operation offload and centralized key management for VNFs that deal with encrypted traffic. CaaS also describes the needs of applications requiring Public Key Infrastructure (PKI) functions, such as VM to VM traffic with IPSec. This scenario describes CaaS as an integrated solution providing centralized key management in virtualized environments, easily scalable to the number of key stores and amount of crypto operations per second.

**#9 Network Slicing**: The key concept of network slicing consists in running multiple logical networks on top of a shared network infrastructure. As a consequence network operators can partition their compute and processing capabilities among applications with distinct purposes. For instance, two applications, *i.e.*, IoT and smartphone, can connect to the same network, the same service provider, but the path and resources each application uses can be different. Although network slicing can be defined on top physical structures, the advent of SDN and NFV eases its implementation because of the characteristics inherent to virtualization, such as elasticity, resource sharing, and dynamic chaining.

**#10 Virtualization of Internet of Things (IoT)**: the IoT encompasses a broad set of services and applications capable of network communication and running on top of physical devices, vehicles, home appliances and others. With the advance of network access ubiquity, IoT has increased the potential of data communication massively, and thus, is one of the leading causes of 5G development. Apart from the improvement of radio and core networks that IoT demands, supporting such evolution requires a variety of network functions to provide control, connectivity, authentication, storage, and routing. In such a context, the virtualization of network functions allows the efficient utilization of IoT services and introduces agility to the development of new services. The proper NFV operation on IoT requires studies on NFV platforms able to integrate applications, analytic utilities, security, and management.

**#11 Rapid Service Deployment**: NFV is likely to be applied in a variety of fields, and one of its primary objectives is the quick deployment and instantiation of VNFs. This use case aims to promote and discuss mechanisms to allow the rapid development of VNFs and as well their deployment in the desired network.

**#12 VNF composition across multiple administrative domains**: the advent of 5G networks is fostering a change in the way communication services have been provided, with a pervasive availability of network access. Key aspects to be addressed in the 5G development include, but are not restricted to, shorter latency, faster Time-to-Market, and better QoE.

As we already discussed, NFV technologies can help in meeting such objectives by offering dynamic network services on-demand, with efficient use of resources. However, in a financial view, the various traffic demands from different kind of network users (*e.g.*, IoT applications, home users, and mobile users) cannot justify the investment needed to deploy an infrastructure that supports a fully-elastic network service. A valuable option to this end is leasing: a 5G operator (the one that offers access to end-users) can hire virtualized networks as well as computing environments to run their VNFs. Thus, this Use Case aims to investigate appropriate ways of composing VNFs across multiple administrative domains dynamically.

**#13 Security as a Service (SecaaS)**: despite the benefits of a more pervasive and ubiquitous network that future technologies and 5G brings, end-users and enterprises become more susceptible to cyber-crimes. Proportionally to the cyber-crime growth is the pursuit of cyber-defense techniques, with both organizations and independent developers seeking for new solutions. In this context, NFV technology allows the development of specific VNFs able to address security issues, and these VNFs can be offered in a SecaaS manner. For example, consider the world is experiencing a global attack, and an independent developer publishes in a marketplace a VNF able to mitigate the issue. As soon as the VNF is available, network operators that are suffering from the attack can acquire the VNF to treat the problem.

Although this Thesis does not aim to address a specific NFV Use Case, it is mostly tailored to the second one (VNF-FG), where the central objective is to investigate how to measure the impact that each VNF has in a Service Function Chain (SFC). The concept of VNF-FG applies in the SFC composition, which allows NFVSPs to offer many-fold objective network services, including providing security, improve performance, implement routing rules, and perform protocol translations.

## 2.3 NFV Service Function Chaining

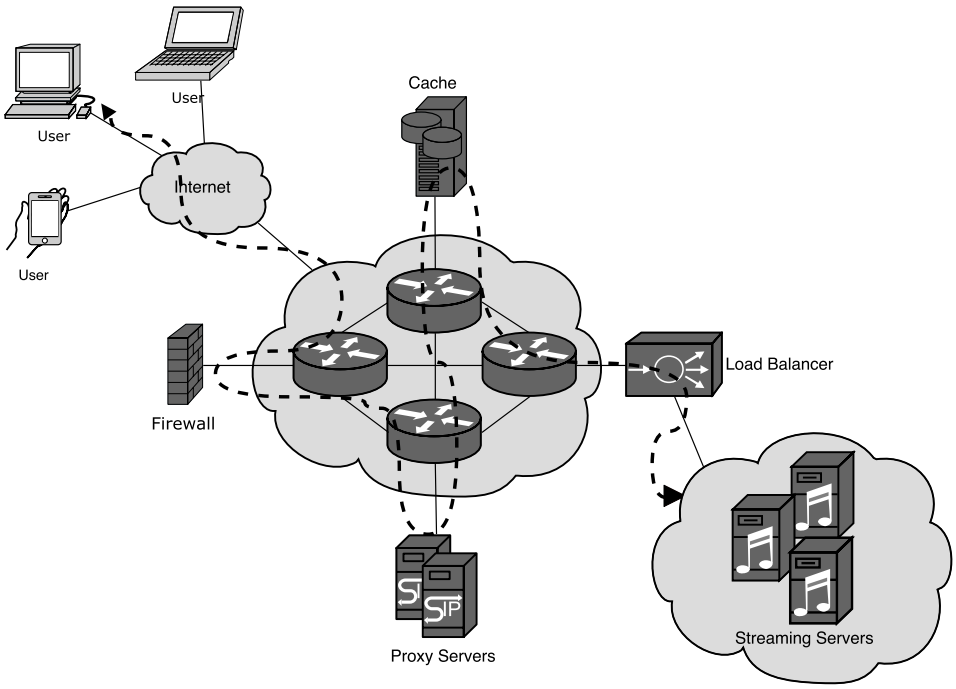
As service providers of the Telecommunications industry take advantage of NFV technology, network services can be provided relying on VNFs, such as Network Address Translation (NAT) servers, load balancers and firewalls. Such VNFs are normally deployed across multiple clouds geographically spread, according to cost, availability or proximity to end-users. In turn, a logical connection is necessary for packets to traverse such VNFs in order to provide a network service. Such interconnection between VNFs is commonly referred to as Service Function Chaining (BHAMARE et al., 2016).

In recent years, the scope of the SFC architecture has reached not only network services, but multimedia and application services as well. Large enterprises, such as banks and global retail stores, also known as Application Service Providers (ASPs) (we also referred earlier to as

ESPs), are relying more and more on SFC to build their services flexibly and cost-effectively. As the number of ASPs that turn to multi-cloud environments grow, the challenges of deploying and managing such SFCs are becoming increasingly complex.

Figure 2.3 depicts an example of SFC. In this example, a music streaming ASP, such as Spotify, uses an SFC to provide security, availability and caching for its users. We can see in the figure that all traffic between the streaming company and the streaming client has to go through proxy servers, a firewall, a cache and a load balancer, connected through four routers, as illustrated by the dotted line.

Figure 2.3: SFC example scenario




Source: adapted from (JACOBS et al., 2018)

One of the management challenges that arise from the SFCs consolidation is the detection of performance problems. As SFCs have multiple VNFs with different natures, each one can have a distinct impact on end-service performance. Also, the service chains may involve multiple NFV service providers, which hinders the ability to correlate individual measurements of VNFs with end-service performance. As investigating how to quantify performance impacts of VNFs in service chains is the central objective of this Thesis, we must understand how these performance issues may arise with NFV. Thus, in the next Section, we discuss details of NFV performance problems.

## 2.4 NFV performance issues

Despite the benefits of NFV and its widespread application in network communication, dealing with performance problems is challenging and critical. The challenge increases for many reasons, including VNFs of diverse types running on top of also different virtualization platforms, VNFs running on multiple administrative domains, and VNFs offered as black-boxes by third-party developers. The criticalness is justified by the characteristic of VNFs intermediating the communication between end-users and end-services, where even small performance problems in a specific VNF can degrade the performance of an entire SFC, consequently affecting QoS and QoE.

In this regard, Wu, He and Akella (2015) argue for at least three categories of issues that can affect NFV service chains performance: bottlenecks, contention, and buggy design/implementation. Bottlenecks occur due to the misallocation of resources to one or more VNFs in the SFC, and for this reason, bottlenecked VNFs burden the regular packet forwarding. When the provisioned resources of a VNF are fewer than needed, the performance of transversing network flows is *bottlenecked*. As VNFs run on top of a virtualization layer (see Figure 2.2), resource allocation (*i.e.*, CPU, memory, network bandwidth, and disk) can be fine-tuned on-demand. However, when allocated resources are under-provisioned, the VNF becomes overloaded and unable to deal with the incoming demand, which may lead to delay in network forwarding or even to packet losses in the worst cases. To detect bottlenecks, operators usually leverage the monitoring of resources at VNF operating system level, with detection models raising flags when resource consumption is superior to a certain threshold. Another approach is the monitoring of QoS metrics, where the solution consists of flagging performance issues when an end-service metric exhibits anomalies (*e.g.*, increased response time or decreased throughput).

Contention takes place when VNFs share resources and their load is superior to the total capacity. When VNFs share limited resources, the network flow performance suffers from *contention*. One of the benefits of running VNFs on top of a virtualization layer is that multiple instances can share underused resources (*e.g.*, memory bandwidth, CPU, and network bandwidth). The counterpart is that when these VNF instances require more resources than the total capacity, they suffer from performance degradation because their requests are enqueued in the contended resources. To diagnose resource contention, NFV operators also leverage the monitoring of operating system level metrics to verify if the total of resources consumed by VNFs is superior to the total capacity of the substrate.

A misconception in the design or implementation of a VNF software may imply in inefficient computation and consequently delay of packets processing. For instance, a TCP accelerator VNF (SIRACUSANO et al., 2016) that optimizes TCP phases to increase performance may wrongly call a bugged TCP queueing method, like the one exposed in the *SegmentSmack* flaw (REDHAT, 2018), leading to a CPU saturation at the VNF. Also, bugs can propagate errors in the service chain datapath, making processing nodes unnecessarily expending computation

resources. For example, consider a university NFV scenario where a packet modifier VNF tags group informations (*e.g.*, faculty, staff, and students) to be used in policy decisions (KANG et al., 2017). After that, tagged packets go to a routing VNF (*e.g.*, SDN controller) to decide by which switches and VNFs each network flow must transverse according to the established policy. Now suppose that the first VNF wrongly writes the source group into the destination field, this could make the second VNF forward the packets through an incorrect path, wasting resources in the entire SFC.

The three kinds of performance problems are similar regarding their reactive nature of diagnosis, which means that when the issue is detected, the SFC performance has already been affected. In another direction, Sauvanaud et al. (2016) argue for anomalies in resource consumption as a distinct type of performance problems. The authors claim that if one or more VNFs exhibit an unusual behavior, it can signalize that an issue is likely to occur. In such context, machine learning mechanisms may apply to recognize whether unusual behaviors are indicative of performance degradation or not.

Given the above discussion about NFV performance problems, in this Thesis, we aim to propose a model able to quantify the amount of performance degradation a VNF causes to the SFC, independently if it originates from under-provisioned capacities, resource contention, buggy implementation, or unusual behavior. Before detailing our proposal, in the next Section, we both position our research in the state-of-the-art of bottleneck identification and performance quantification in NFV and discuss the related-work.

## 2.5 State-of-the-art

Some research efforts have been carried out to address the two research areas of this Thesis in the NFV context: bottleneck detection and performance quantification. Table 2.1 summarizes the research efforts that address these areas. Below, we discuss the approaches and metrics used by each work in the state-of-the-art. For the sake of organization, we divide the discussion according to the central objective of research works. We leave the review of state-of-the-art limits to Section 2.6.

### 2.5.1 Bottleneck detection

In a preliminary initiative to detect resource contention-related NFV performance degradations, we proposed a *threshold monitoring* approach (PFITSCHER et al., 2016). The central idea consists of monitoring OS-level metrics and performing threshold comparisons to establish resource provisioning status levels (*i.e.*, under-provisioned, correctly provisioned and over-provisioned). To that end, two operating system metrics were taken into account: CPU and steal usage. Regarding the performance diagnostic, the monitoring agents continuously collect metrics and compute a time-series average, defining VNF states according to the thresholds.

Table 2.1: State-of-the-art methods and metrics

| Reference | Objective | Method | Metrics of the model |
|---|---|---|---|
| (PFITSCHER et al., 2016) | Bottleneck detection | Threshold monitoring | CPU usage, and Steal time |
| (WU; HE; AKELLA, 2015) | Bottleneck detection | Customized metric | Network queuing, and packet loss |
| (NAIK; SHAW; VUTUKURU, 2016) | Bottleneck detection | Customized metric | Packet processing time, and throughput |
| (GEMBER et al., 2013) | Bottleneck detection | Threshold monitoring, and iterative provisioning | Packet processing time, and CPU usage |
| (CAO et al., 2015) | Bottleneck detection | Threshold monitoring, and iterative provisioning | CPU usage, NIC usage, and memory usage |
| (GALLARDO; BAYNAT; BEGIN, 2016) | Performance quantification | Analytical model | CPU usage, and arrival rate |
| (PRADOS-GARZON et al., 2016b) | Performance quantification | Analytical model | arrival rate, and service rate |
| (SUKSOMBOON et al., 2016) | Performance quantification | Analytical model | CPU usage, NIC usage, cache misses, and arrival rate |
| (SAUVANAUD et al., 2016) | Performance quantification | Pattern recognition | Black-box, and gray-box |
| (KOIZUMI; FUJIWAKA, 2015) | Performance quantification | Analytical model, and statistical regression | CPU usage, NIC usage, and memory usage |

Source: the Author

Finally, when the VNF experiences a state change, the monitor triggers an event to warn a manager about the new VNF state.

Cao et al. (2015) propose the NFV-VITAL framework for performance characterization of general NFV scenarios. The approach combines both *threshold-based* and *iterative resource provisioning* methods by stressing the chain with distinct workloads and VNF sizes, monitoring resources in each VNF, and comparing thresholds to detect bottlenecks. In fact, the central objective of the paper is to characterize the performance of an NFV IMS implementation. The paper leverages multiple resource usage metrics for the characterization, including, CPU, memory and network interface card.

Gember et al. (2013) also combine iterative *resource provisioning* with resource monitoring to search for bottleneck VNFs. A bottleneck is pointed-out if the processing time of packets increases in a time window or if memory and CPU usages exceed a threshold. Also, an approach for scaling up and down resources of each VNF is applied to observe whether the provisioning action resulted in improvements or deterioration of the overall chain performance.

In a distinct direction, two works propose *customized metrics* to address the bottleneck detection problem. Wu, He and Akella (2015) introduce the use of network queueing and packet loss observations to detect resource contention situations. Their solution consists on instrumenting VNFs to measure resources' queues, pointing out bottlenecks on specific resources. Similarly, Naik *et al.* (NAIK; SHAW; VUTUKURU, 2016) uses packet processing times and throughput of VNFs to detect bottlenecks. By mirroring VNFs' network interfaces, a central management instance computes and learns the performance profiles of each VNF. Then, if an abnormal behavior occurs in one instance, the bottleneck is pointed out.

### 2.5.2 Performance quantification

Gallardo, Baynat and Begin (2016) propose an *analytical model* for the performance evaluation of virtual switches. Although switches are network appliances intrinsic to the network communication, virtual switches have been considered VNFs because of their flexibility in resource allocation. In the proposed solution, RX queues of each vNIC and each CPU core are modeled as the elements that compose a virtual switch. Then, the performance analysis is applied to the underlying computation of the global system, regarding CPU core utilization, loss rate, sojourn time of packets, and buffer occupancy. The main strategy of the model is to decompose the system into a set of independent queuing models, and to introduce service vacations models. Vacations refer to the in-between packets time, *i.e.*, the time where the server was collecting the next packet from the RX queue before processing it on CPU. Through measurements of arrival rates, the model uses probabilities to estimate the missing parameters and, as a consequence, estimate the global system performance. Likewise, Prados-Garzon et al. (2016b) propose a theoretical framework to evaluate the performance of a special purpose VNF of LTE networks: the MME. The proposal leverages a queueing network model that uses arrival rates and system service rates to assess the mean system response time.

Suksomboon et al. (2016) also use *analytical modeling* to predict the performance of software routers. The authors propose predicting the maximum throughput a software router can achieve for packet forwarding, according to its allocated resources. The paper considers three factors as influencing in the performance: Ethernet bandwidth, CPU speed and shared Last Level Cache (LLC). Instead of using queueing theory, the model for estimating VNF throughput is based on a mathematical formulation that includes cache contention as a factor of performance degradation. Cache contention occurs when *corunner* VNFs compete for the same core, resulting in cache misses. In turn, Sauvanaud et al. (2016) propose a solution for anomaly detection and root cause identification based on *pattern recognition* techniques. The proposal combines black-box and gray-box monitoring data with machine learning mechanisms to detect anomalous situations. The root cause of problems detection derives from a database trained with both monitoring metrics and SLA violations. A detection model compares the behavior of VNFs through a knowledge database to result in a classification probability. Next, a probability threshold defines whether a VNF is performing as an anomaly: if true, such VNF is probably the root cause of problems.

Koizumi and Fujiwaka (2015) combine *statistical regressions* and *analytical modeling* to develop a model for estimating NFV service chain performance. The proposal takes into account the *processing overlap* to predict the execution time. Processing overlap is the time taken by an execution node (*i.e.*, VNF) to process distinct services simultaneously. The performance estimation process integrates a Colored Timed Petri Net (CTPN) model and statistical regressions to compute the node's processing time dynamically. In such process, historical information from CPU, memory, and network usage are applied in multiple regression models to estimate

parameters of the mathematical formulations.

## 2.6 Chapter summary and remarks

In this chapter we reviewed the NFV central concepts and discussed the state-of-the-art on bottleneck detection and performance quantification. First, we introduced the general idea of Network Function Virtualization, which consists of running previous hardware-dedicated middleboxes in softwarized implementation. Also, for the sake of exemplification, we describe the most recent Use Cases of NFV proposed by ETSI. Then, we apprise the concept of service function chaining, where VNFs are logically chained to compose network services. Such concept is a milestone for this Thesis, since our main objective is to quantify the likelihood of a VNF burdening the SFC performance. Next, we finish the background of this Thesis by discussing the main performance problems investigated in the NFV literature.

After providing an overview of NFV background, we analyzed the state-of-the-art on bottleneck detection and performance quantification in NFV service chains. In such analysis we notice that different metrics (traditional and customized) have been considered for bottleneck detection and performance quantification in NFV. A set of major concerns arise when solving these problems simultaneously: (*i*) threshold monitoring approaches (PFITSCHER et al., 2016) (GEMBER et al., 2013) (CAO et al., 2015) are not able to quantify performance degradations and they require a calibration step to accurately diagnose bottlenecks; (*ii*) customized metrics depend on low-level instrumentation of operating system queues (WU; HE; AKELLA, 2015), or need access to network devices to perform mirroring (NAIK; SHAW; VUTUKURU, 2016); (*iii*) iterative provisioning can delay the bottleneck diagnosis process, and fail to quantify degradations; (*iv*) analytical models (GALLARDO; BAYNAT; BEGIN, 2016) (SUKSOMBOON et al., 2016) (KOIZUMI; FUJIWAKA, 2015) require a deep knowledge of each type of VNF and their internal communications; and (*v*) pattern recognition and statistical regression need a large set of results before providing precise estimates of performance. Nonetheless, the set of available metrics and VNF types are heterogeneous, increasing the complexity, for operators, to decide which metrics they must monitor. We argue in favor of an approach based on a customized metric integrated with an *online* learning mechanism able to weigh the importance of distinct resources in each VNF type.

In this Thesis we propose the `guiltiness` metric to fill the presented gaps, allowing to quantify the performance impacts of individual VNFs in general service chain scenarios. In the next Chapter, we perform experimental evaluations to investigate the ability of metrics from literature to express NFV performance. Thus, by revealing which metrics are more representative we address the first research question (RQ-1).

# 3 NFV PERFORMANCE CHARACTERIZATION

There is a significant amount of metrics available to characterize the performance of individual VNFs in an NFV service chain. When network operators want to assess the performance of VNFs on forwarding network flows, they usually monitor performance metrics. These metrics include resource usage levels (*e.g.*, CPU, memory, network, and storage), key network performance indicators (*e.g.*, jitter, delay, latency, throughput, and packet loss), operating system indicators (*e.g.*, cache hit/miss, CPU speed, logs, and resource capacities), and special purpose metrics (*e.g.*, response time and deadline misses). However, there is no consensus of neither what metrics operators must observe, nor the accuracy of these metrics on reflecting performance degradations in SFCs. Establishing the limits of the metrics' ability to quantify degradations is the objective of the first research question. As the first step to this end, we review the literature to define what are the metrics used by researchers to evaluate their proposals. Next, we perform a straightforward analysis of these metrics, through experimental evaluations in NFV scenarios.

## 3.1 Survey methodology

To survey the literature and find out metrics used in NFV evaluations, we develop a search query containing the keywords that delimit our research area. We explain our assumptions on search query development in Section 3.1.1. Then we use a four-phase process to perform the search and refine results: *execute*, *classify*, *filter*, and *merge*. Section 3.1.2 discusses each of these phases.

### 3.1.1 Search query

We use Scopus (SCOPUS, 2018) database as the search tool to survey NFV research results. The reason why we use Scopus is that it includes sources from other relevant databases in computer networks field, such as the IEEE Xplore (IEEE, 2018) and The ACM Digital Library (ACM, 2018). In Scopus, the "advanced search" tab provides operators and field codes to allow users to formulate a string query. Our search query was built targeting research papers that both address NFV challanges and present an evaluation of their proposals. To that end, we split the query into three parts. The first part delimits the scope of research. The second part filters search results to include evaluations. The last part restricts the results to guarantee the presence of at least one well known evaluation method (JAIN, 1990). The resulting query is as follows:

```
(TITLE-ABS-KEY("network function virtualization")) AND
(ALL(evaluation) OR ALL(scenario)) AND
(ALL(experimental) OR ALL(simulation) OR ALL(emulation) OR ALL(analytical))
```

To narrow the query result on papers that focus on solving NFV problems, we include the phrase "*network function virtualization*" to be found in title, abstract or keywords through the operand TITLE-ABS-KEY. In the second part of the query, we establish that resulting paper must include one of two words: *evaluation* or *scenario*. By establishing the second part of the query, we aim to search for papers that use these terms in sections' names or some specific context inside the text. The third part restricts results to contain at least one of the well-known methods of performance evaluation (JAIN, 1990): *experimental*, *simulation*, *emulation*, and *analytical*. It is worth noticing that the operand ALL will search for the word in all the text, including references, which requires a refinement process (explained below).

### 3.1.2 Refinement process

We leverage a refinement process to define the state-of-the-art of NFV evaluation scenarios. This process consists of four phases. First, we execute the research query computing a list of papers. Next, we filter results excluding works according to a established criteria. Then, we classify papers according to a taxonomy (objective, metrics, and evaluation scenarios). Finally, we rerun the query, merge new results with previous papers, and perform filter and classification. Each of the four phases are described as follows:

- *Execute* - We continuously execute the query to collect articles in the literature. The first run was on March 08, 2017 and resulted in 143 articles. The second run was on March 29, 2017, resulting in 192 articles.
- *Filter* - We exclude papers that could misrepresent the view of research on NFV. We skimmed each paper on the list and dropped the ones that: are survey-like and do not assess any aspect of NFV solutions, do not propose any NFV solution, or only provide case studies to evaluate their proposals.
- *Merge* - After each query execution, we merge new results. In addition to automatic software verification through Mendeley (MENDELEY, 2018), we verify papers individually to exclude duplicated entries. The number of papers in the list remained on 104 after all iterations of executing, filtering and merging phases.
- *Classify* - Finally, we classify each paper regarding the monitoring metrics and evaluations scenarios. To do so, we first read the abstract of each paper to determine the research context. Then, we analyze the papers' evaluation section to identify characteristics of proposals assessment.

## 3.2 Literature overview

In the following sections we present an overview of the literature regarding the *metrics* monitored in NFV papers (Section 3.2.1) and the *scenarios* that researchers used to evaluate

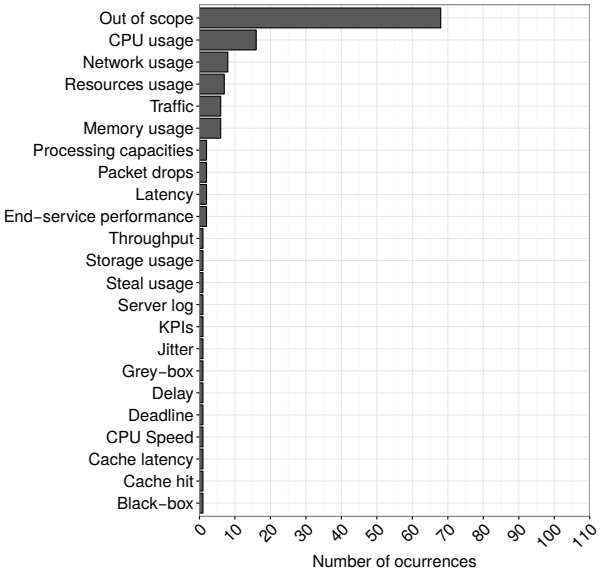
their proposals (Section 3.2.2).

### 3.2.1 Monitoring metrics

We found, in the literature survey, that works on NFV monitoring aim to solve two main research objectives: elasticity management and performance diagnosis. In the former, the NFV orchestrators rely on metrics measurements to scale in/out VNFs, or their resources, on demand. In the latter, NFV managers combine environment reports with diagnosis methods to detect performance issues (which is the central objective of this Thesis). In addition to the metrics found in the NFV literature survey, we also include for analysis the reference metrics presented in the state-of-the-art discussion (Section 2.5). Figure 3.1 accounts, for each metric, the number of citations (Fig. 3.1(a)) and the respective citing authors (Fig. 3.1(b)).

Figure 3.1: Monitoring metrics in NFV literature

(a) Distribution




(b) Citing authors


Source: the Author

As Figure 3.1a shows, most of the works do not provide information about monitoring metrics (denoted as 'out of scope'), as these works neither perform elasticity management nor address performance diagnosis issues. Such works propose solutions to various areas, including resource allocation, security, and management policies. The remaining metrics found in the literature follow a long-tailed distribution which we split into four categories: *network*, *memory*, *processor*, *I/O*, and *service-related*. Below, we examine these metrics and their potential for bottleneck identification.

Most of the metrics exploited in NFV research are *network-related*: NIC throughput, NIC

usage, NIC queueing, packet drops, latency, jitter, and traffic patterns. NIC throughput denotes the amount of data, or amount of requests, that one VNF can process in a time interval (MENASCE et al., 2004). The intuition is that the VNF with the highest throughput in the chain probably retains a higher incidence of network flows. Hence, such VNF is more likely to impact the SFC performance when stressed. NIC usage, derived from NIC throughput, computes the rate between NIC throughput and the link bandwidth over time. The greater the NIC usage in the VNF, the greater the chance of the VNF being a bottleneck, because the NIC is close to its saturation. On the flip side, if a VNF suddenly exhibits a NIC usage fall, it could also be a signal of a performance anomaly caused by a contended resource.

When the NIC becomes saturated, it is subject to requests queuing. NIC queueing occurs in two situations: when the VNF's transmission demand is higher than the available bandwidth, datagrams are enqueued on the output buffer of the virtual network interface; and when the reception rate is greater than the processing throughput, queueing occurs in the input buffer (JACOBSON, 1988; PFITSCHER; PILLON; OBELHEIRO, 2013). Intuitively, if one detects that there is queuing occurring at a node, then it may be a bottleneck. Packet drops occur just after queueing when a buffer is full, so incoming requests are dropped. If packets are being dropped in a given VNF, it is because one of the resources inside the VNF is saturated, and thus, the VNF may be a bottleneck (WU; HE; AKELLA, 2015).

Network latency and jitter have also been used in the NFV literature to evaluate the communication between VNFs. The network latency measures the time taken by a packet to go from a VNF to the next VNF in the chain. Usually, network latency is also referred to as the round-trip delay time (RTT), which is the measure of time packets take to go to the destination node and the source node receiving the delivery acknowledgment. Regarding bottleneck identification, by comparing latency values among all VNFs of a service chain, one can identify which links are congested, and, hence, are responsible for delaying the forwarding of requests to the end-service. Jitter is the average deviation from the mean latency. It accounts for the amount of latency variation over time. Some applications are sensitive to high jitter, and thus, it is essential to identify which link, or VNF, is increasing the overall jitter observation. The characterization and recognition of traffic patterns have also been used for bottleneck identification. The analysis of network traffic pattern allows operators to foresee load peaks situations in each VNF of the chain and deal with them accordingly.

Concerning *memory-related* metrics application for VNFs' diagnosis, the NFV literature survey resulted in three metrics: resident percentage, committed percentage, and cache hit/misses. The resident memory represents the amount of memory that is effectively in use by the running processes and the operating system. Resident memory is given by the subtraction of free memory, cache and I/O buffer from the total physical memory. In turn, the committed memory expresses the amount of memory allocated for a given process (PFITSCHER; PILLON; OBELHEIRO, 2014). As applications do not use all the requested memory, modern operating systems reserve more than the memory available in the physical machine, and thus, it is common to

observe more than 100% of committed memory. Hence, if a machine presents high usage percentages on these metrics, the VNF's performance decreases, which can result in a flow contention. In addition to memory usage, statistics about cache hits/misses and cache latency can provide meaningful information about the status of VNFs. When a system exposes a high rate of cache misses, this could indicate a cache contention situation, and thus, the VNF can be a bottleneck in a service chain (SUKSOMBOON et al., 2016).

Among *processor-related* metrics, which are most commonly resorted to as performance indicators, we analyzed the four most prominents: CPU usage, steal usage, CPU speed, and active percentage time. The CPU usage represents the percentage of time that a processor is busy during a time interval. Two aspects suggest an investigation about CPU usage. First, CPU occupation is a signal of resource stress: the higher the usage, the greater the likelihood that the system cannot support an increasing demand. Second, using this metric, alongside the network throughput, one can predict the VNF's service demands through laws from analytical models, such as the Utilization Law (GUNTHER, 2007; MENASCE et al., 2004). The steal time metric represents the portion of time that a virtual machine waits to run its processes while virtual CPU was not allocated due to physical resource contention (RIEL, 2006). The reasoning behind this metric is that if steal occurs, the performance of the host decreases, which in turn affects the VNF too. The CPU speed refers to the nominal capacity of the processor provisioned to the VNF. The meaning of the metric for bottleneck detection depends on the amount of this capacity the VNF is using, which is strictly related to CPU usage. Finally, active percentage time refers to the average amount of time a given resource — in this case, CPU — is active in an observed time series. This metric is derived from the active duration time metric (WANG; ZHAO; ZHENG, 2005), which accounts for the period of time that a resource is in a non-idle state. Thus, regarding bottlenecks, the more active a node is, the higher is its performance impact on the service chain processing result.

The monitoring of *service-related* metrics also has a potential for signaling chain performance issues. The entries we found in the literature for this kind of metric are as follows: arrival rate, service time, response time, server logs, key performance indicators (KPIs), and deadlines achieved. The service arrival rate accounts for the number of requests a certain node receives by the unit of time. The intuition is that the VNF with the higher arrival rate in the service chain is the busier one, and so, among the others, it is the more likely to be a bottleneck. Service time (also known as *sojourn* time) corresponds to the time taken by a request to be processed in a node. The VNF that has the longest service time in a chain is the one that most delays the overall chain forwarding process. Response time usually refers to end-services performance. This metric explains the entire space of time consumed by a request in an end-service, including queueing wait time and processing time. Some works, as the proposal of Gember et al. (2013), monitor the response time of end-services to find out optimal resource allocations and prevent bottleneck occurrences. The analysis of server logs is a useful source to extract data for machine learning algorithms. Such information includes the number of users logged, errors in

operating system charging process, invalid memory address references, and I/O driver breaks. Similarly to server logs, the observation of several KPIs allows an intelligent computer system to both classify anomalous behaviors and detect the root cause of problems. The term KPIs has a broad meaning and can cover an extensive set of metrics, including the ones we discussed throughout this section. The same occurs for the metrics referred to as gray/black-box: the first term designate to metrics measured with the instrumentation of the monitored target, and the second, with measurements collected indirectly (*i.e.*, outside the monitored target). Finally, the number of achieved deadlines provide insight on how well the system is handling time-sensible requests. Li, Phan and Loo (2016), modeled the set of requests a chain receives as each one having soft-real time requirements. Thus, by combining real-time scheduling algorithms (*i.e.*, Earlier Deadline First) and integer linear programming, the solution proactively provision VNFs so that all packets meet their deadlines.

In addition to the previous metrics, we analyzed one *I/O-related* metric. Neither the state-of-the-art (Section 2.5) nor the works in the literature survey consider I/O-related metrics in their analysis, and, in our view, the reason for I/O exclusion is mainly because VNFs are essentially network packet forwarders, which means that they receive packets, process them, and forward to the next node; such process does not require storing information about network flows. However, we argue that several VNFs rely on I/O operations, such as when an Intrusion Prevention System (IPS) access a database to detect malicious flows, or when an IMS component registers incoming calls. Thus, we include the I/O operations per second metric in our analysis. Instinctively, the behavior of this metric denotes that the closer this resource comes to its limit, the higher the probability of it causing a bottleneck.

Given this huge spectrum of research, we review the basis behind these metrics to reduce the search space as a way of having a more straightforward analysis. We disregard *processing capacities*, *traffic patterns*, and *server logs* in assessments because these metrics do not offer quantitative information for comparing the VNFs health. We also abdicate of evaluating *gray/black box* and *KPIs* since these terms are unclear about which metrics they include. The cases of *deadline achieved*, *CPU speed*, *cache hit/miss*, and *cache latency* are also set aside as these metrics require specific instrumentation to be monitored, and therefore, are difficult to generalize. Metrics that apply to measure the performance of network communication between VNFs (*i.e.*, latency and jitter), where also disregarded because they are useful to indicate problems in the infrastructure (*e.g.*, switches and routers), and not in the VNF itself. Considering this refinement, we converge our study in the analysis of the following metrics:

- *network-related:* throughput, NIC usage, NIC queueing, and packet drops;
- *memory-related:* resident memory, and committed memory usages;
- *processor-related:* CPU usage, steal usage, and active percentage time;
- *service-related:* response time, arrival rate, and service rate; and
- *io-related:* I/O transactions per second.

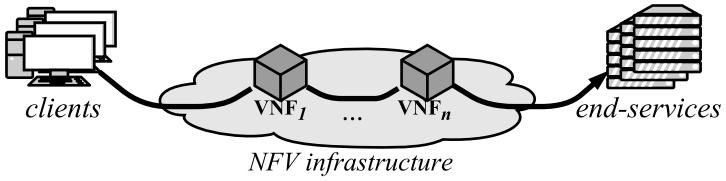
### 3.2.2 Evaluation scenarios

After defining the promising metrics for pointing out bottlenecks in network function service chains, we now discuss the footprint of evaluation scenarios in NFV. We characterize these scenarios in terms of three aspects: chain ordering (Figure 3.3a), chain size (Figure 3.3b), and categories of VNFs (Figure 3.4).

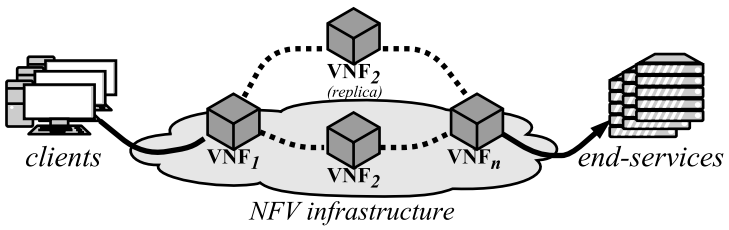
In an NFV scenario, as exemplified in Figure 3.2, flows from clients to servers pass through $n$ VNFs that form a service chain. The order of VNFs a given traffic flow pass through are of two types: *sequential* or *parallel*. In sequential chaining (Figure 3.2a), traffic flow goes through all VNFs in a single branch. In parallel chaining (Figure 3.2b), traffic splits into multiple branches of VNFs. Divisions in a chain occur for performance reasons, where multiple replicas of VNFs run to enhance forwarding and processing capabilities. In fact, the parallel chains also have a sequential part, could be then denoted as a hybrid chain. The results from the survey (Figure 3.3a) show that most of the works use one of these two categories: around 70% of papers exhibit evaluations through sequentially chained VNFs and 25% use VNFs chained in parallel. The remaining 5% of papers split into one of this two categories: either they do *not provide* information about chaining order; or, chaining is *out of scope* of their research (*i.e.,* only assess performance aspects of individual VNFs).

Figure 3.2: NFV ordering scenarios

(a) Sequential chaining



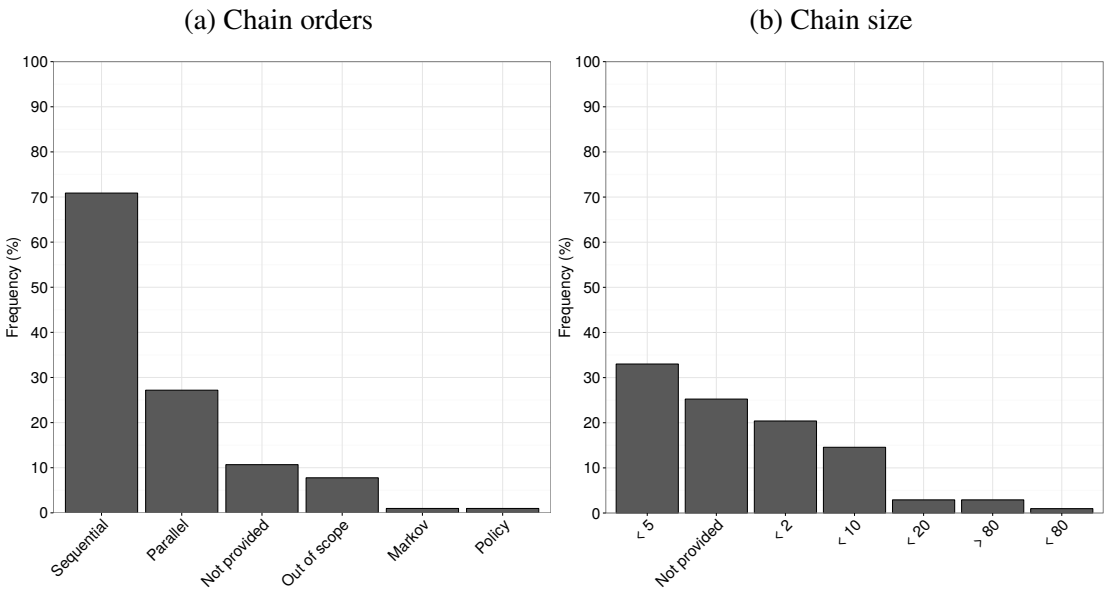

(b) Parallel chaining




Source: the Author

NFV evaluation scenarios are also distinguished by the size of the chains. The number of available VNF types allow researchers to perform tests in distinct scenario sizes, varying from small-chains (less than 5 VNFs) to big-chains (up to 80). As Figure 3.3b depicts, the major-

Figure 3.3: NFV literature chaining characterization



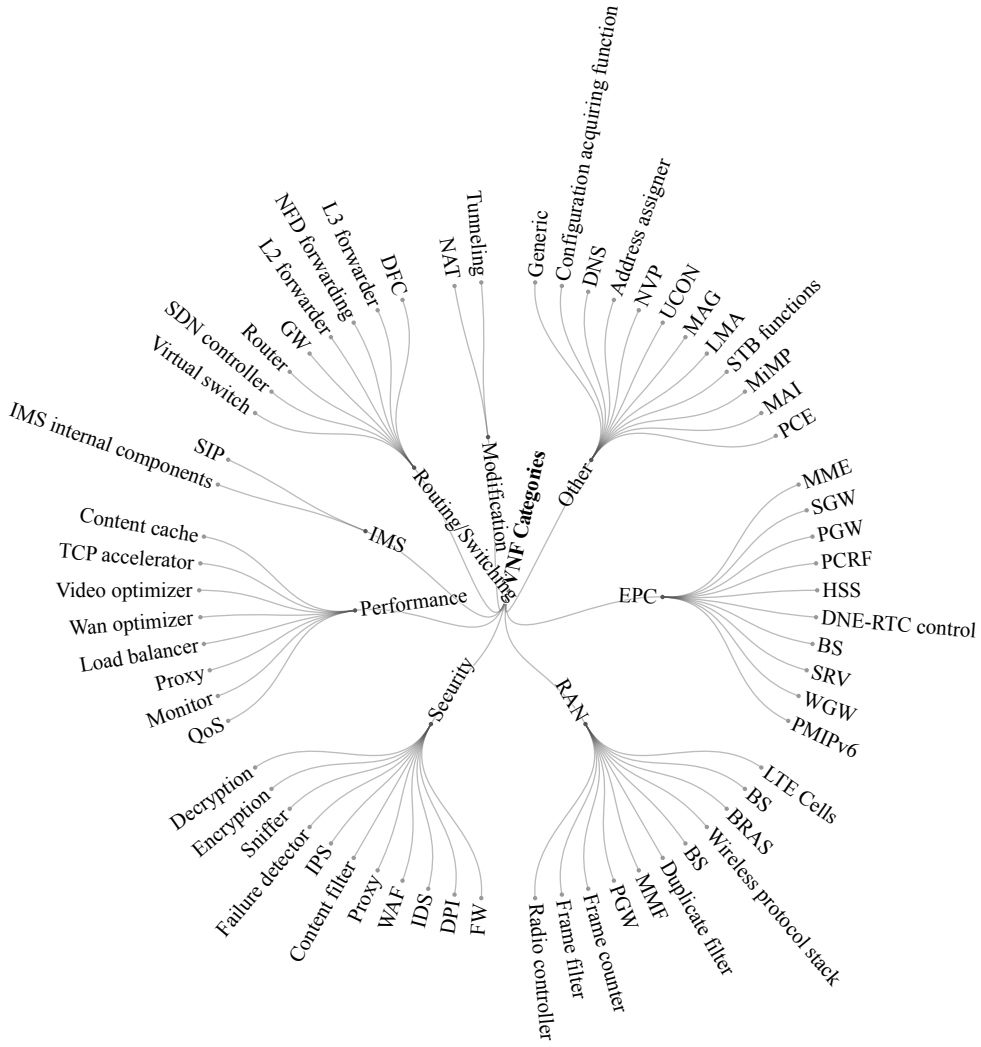

Source: the Author

ity of works use up to 10 chained VNFs and, among these cases, most use up to a maximum of 5. As we delve into these results, we notice that the chain size depends mainly on evaluation method and order. While small scenarios usually refer to experimental evaluations and sequential chains, larger scenarios appear in simulations and parallelized chains.

A plethora of VNFs have been used during NFV development and consolidation. As software capabilities allows to decouple network functions from special purpose hardware-defined middleboxes, researchers have been proposing a spread amount of VNF types. This diversity of network functions is due to the different goals in which they were mainly designed for, such as providing security, network traffic steering, performance enhancement, and so on. We explore this diversity in the NFV evaluation scenarios and classify these VNFs according to eight categories. Figure 3.4 depicts these categories and the VNFs each one includes. Below we describe each of these categories and the related VNFs, details about each VNF functioning are provided in Appendix B.

- **Security** - Security-related network functions are broadly known and utilized. The goals of this type of network functions include: providing protection against intrusions, *e.g.*, Intrusion Prevention Systems (IPSes), and Intrusion Detection Systems (IDSes); filtering malicious packets, *e.g.*, firewalls, Deep Packet Inspection (DPI), sniffers, Web Application Firewalls (WAFs); and preventing failures, *e.g.*, Failure Detectors. In addition, network functions that perform packet encryption, decryption and proxying were also included in this category.
- **Routing/Switching** - Routing and switching functions deal with the process of selecting

Figure 3.4: VNF categories and types included in each




Source: the Author

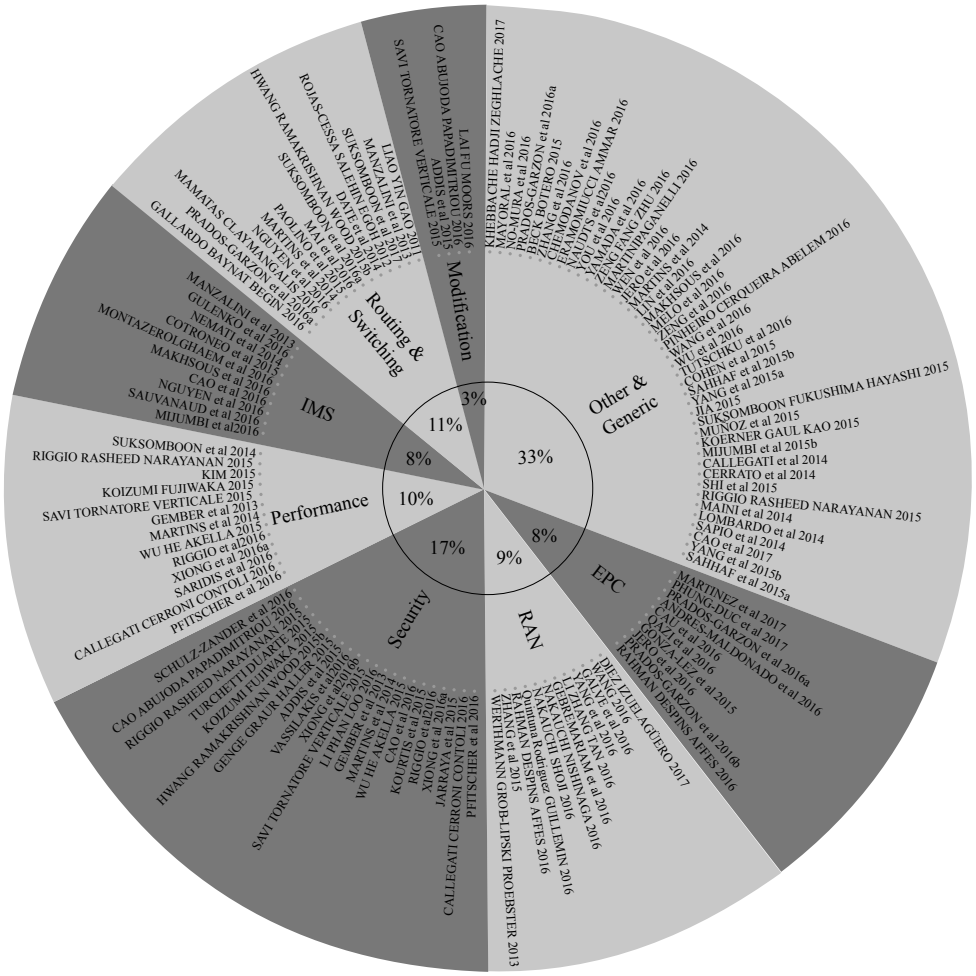
the path that a packet will traverse inside the network or across different networks. We have included in this category, routers, virtual switches, software routers, L2 and L3 forwarders, and any type of network function that only forwards or steers packets.

- **Packet Modification** - Functions often modify packets inside a network, either to translate or assign addresses, such as address assigners and NAT functions. Moreover, tunneling functions do not necessarily modify packets, but they wrap the packet inside another packet, thus we have included them in this category.
- **Performance** - Functions that aim to improve Quality of Experience (QoE) or Quality of Service (QoS) of users are included under the scope of this category. We have encountered, in the literature overview, functions that relate with these two goals, *e.g.*, content caches, load balancers, video optimizers, TCP accelerators, network usage controllers, and WAN optimizers.

- **Evolved Packet Core** - The Evolved Packet Core (EPC) is a framework that was standardized by the 3rd Generation Partnership Project (3GPP) and is divided into functions that are virtualized to validate 4G solutions in next generation mobile scenarios. The framework, and thus this category, comprises four functions: MME (Mobility Management Entity), SGW (Serving Gateway), PGW (PDN Gateway), and HSS (Home Subscriber Server).
- **Radio Access Networks** - Radio Access Networks (RANs) allow the communication between single devices and different networks taking advantage of radio technologies. Within the scope of RANs, some functions such as BaseBand Unit (BBU), Base Station (BS), radio controller, LTE Cells, wireless protocol stack, WiFi GK, and Wireless GateWay (WGW) are commonly used in assessing proposed solutions.
- **IMS internal components** - The IP Multimedia Core Network Subsystem (IMS) internal components are classified in this category. These functions include services for controlling multimedia IP calls through SIP protocols, *e.g.*, Call Session Control Function (CSCF), Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF), and Serving-CSCF (S-CSCF).
- **Other functions** - Some specific functions are not related to any of the aforementioned categories, and thus, we classify them as *other functions*. These network functions perform filtering, packet counting, name resolution (DNS), real-time communication control (DNE-RTC Control), and connectivity to end-users through STB functions (Set-Top boxes). There are also generic functions in this category, which are developed without a specific purpose. The function of this kind of VNF is to simply forward flows through the sequence.

Besides understanding the types and categories of VNFs it is important to have a broad view of functions purposes. To establish a base scenario for evaluations, is mandatory to account for the distribution of VNFs usage. Thus, we analyze the proportion of categories in NFV evaluation scenarios literature. Figure 3.5 depicts the distribution of VNF categories on the surveyed papers and the respective citing authors. Most of researchers leverage generic functions in performance assessments and management/orchestration issues. Hence, 33% of evaluation scenarios use VNFs of *other functions* category, in which generic VNFs represent 30% of the total. The second most used category of VNFs is of *security* type, representing 17% of the total, in which the firewall by itself accounts for 15%. From the third most used category to the seventh there is a similar distribution: 11% use Routing/Switching, 10% use performance-related VNFs, 9% of scenario considered the RAN functions, 8% use EPC functions, and also 8% use IMS internal components functions. Finally, we found that it is not so common to use *Packet Modification* functions in NFV evaluations, as only 3% of papers use VNFs from this category.

Figure 3.5: VNF categories in NFV literature and citing authors




Source: the Author

## 3.3 Representativeness assessments

After the characterization of NFV evaluation scenarios regarding metrics, chains, and VNFs, in this section we establish base scenarios and evaluate the representativeness of this metrics on quantifying end-service performance. The representativeness assessments aim to address the first research question:

**RQ-1:** *Given that the diagnosis of NFV performance has to take into account VNFs' particular characteristics, what is the set of representative metrics able to quantify the performance impacts imposed by each VNF in a service chain?*

To provide RQ-1 with an answer we first summarize survey results and define the two scenarios will serve as base to the experimental assessments, in Section 3.3.1. Then, in Section 3.3.2, we analyze how measurements of each metric pointed on Sec. 3.2.1 behaves against workloads executed on the base scenarios.

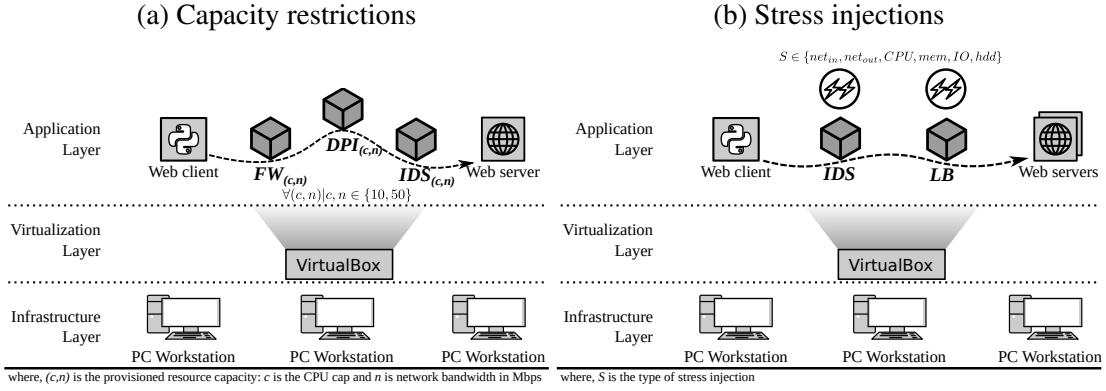
### 3.3.1 Base scenarios

Given the findings from the literature survey, we define two base scenarios: one sequential with security VNFs (*i.e.*, firewall, DPI and IDS) and one parallel with the IMS internal components. We choose security functions in the first case because it is the most used category with a specific purpose (generic functions can be anything that forward packets). In the second case, we select IMS internal components because of two reasons: first, there are public available implementations of parallel service chains (CLEARWATER, 2018), which eases the experimentation process; second, IMS implementations have also been used in the evaluations provided by the state-of-the-art solutions (Section 2.5), making the results comparison fairer.

#### *3.3.1.1 Base scenario 1 - sequential security VNFs*

Three works (PFITSCHER et al., 2016), (GEMBER et al., 2013), and (KOIZUMI; FUJIWAKA, 2015) performed experiments on a *security-based* scenario, with resource capacity restrictions in VNFs. Our first scenario (Figure 3.6a) reproduces these evaluation efforts, and consists of three security VNFs (*i.e.*, firewall (FW), DPI, and Intrusion Detection System (IDS)) with two levels of network capacity (*i.e.*, 10 Mbps and 50 Mbps) defined by the `tc qdisc` tool from Linux. To observe the behavior of metrics, we submitted an increasing demand of Web requests passing through the service chain. Hence, at a specific timestamp, distinct VNFs can be a bottleneck of the Web server performance. The amount of performance degradation and in what level of demand it occurs vary according to the levels of previous factors: inspection depth and capacity. The deeper the analysis of packets, the higher is resource consumption, and as a consequence, the delay in processing requests. Similarly, as the provisioned network capacity gets more restricted, the end-to-end throughput decreases.

Figure 3.6: Base scenario 1 - Security VNFs




Source: the Author

This scenario runs on top of standard PC workstations with 8 CPU cores and 16 GB of memory, running Debian 9.3 (Stretch) with Linux kernel version 4.9. VirtualBox was used as the hypervisor for the Virtual Machines (VM) that run clients, servers, and VNFs. Each VM runs an Ubuntu server 16.04, Linux kernel version 4.4, and was configured with a single core and 512 MB of memory, except for the IDS that has 1 GB of RAM. Static routes are used to forward the traffic from clients, passing through the VNFs, to the servers. We use the application throughput as end-service metric.

In addition to using capacity restrictions to generate bottlenecks, we evaluate how metrics behave when VNFs are subjected to stress injections in multiple resources (Figure 3.6b). By doing so, we reproduce the evaluation scenario conduced by Wu, He and Akella (2015) in their experiments. In this scenario, the NFV service chain consists of an IDS (implemented through Suricata[1]) and a proxy Load-Balancer (LB) (implemented through Balance[2]), sequentially ordered. During a 260 seconds interval, a Web client submits a constant demand of 30 reqs/s for a 1 MB file hosted in two replicated Web servers. Also, the workload combines periods of regular traffic, where VNFs only process the Web demand, and periods of resource contention. For the latter, synthetic programs saturate the VNFs with: inbound flood traffic, outbound flood traffic, CPU computation, memory allocation/writing, I/O bus writing, and HDD writing. The first stress load is inflicted to the IDS, starting at the 10 seconds mark, lasting for 10 seconds, and repeating every 20 seconds stressing another resource. From the 130 seconds and onwards the LB is subjected to the same stress loads, in the same time frames. Regarding system configuration, we use the same standard PC workstation and VirtualBox images as in the scenario with capacity constraints.

#### *3.3.1.2 Base scenario 2 - IMS internal components*

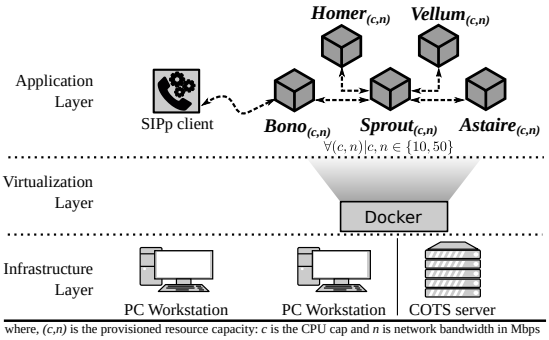
The second scenario used to assess metrics is an implementation of the internal components of an IMS, as used for the evaluations presented in (CAO et al., 2015), (SAUVANAUD et al., 2016), and (NAIK; SHAW; VUTUKURU, 2016). The benefits of elastic resource scaling and dynamic service chaining have propelled the adoption of NFV for IMS internal components. In one of these adoption efforts, the Clearwater project[3] published an open-source implementation of the IMS core standard (ETSI, 2013). Clearwater takes advantage of a load-balancing DNS to provide dynamic horizontal scaling for all internal components, which aids the deployment of evaluations scenarios that require parallel chaining.

Figure 3.7 depicts the Clearwater architecture and its relation with clients and the NFV infrastructure. In such scenario, IMS clients make multimedia calls (commonly VoIP) to communicate through the Internet. For these IP-based communications to be successful, traditional

[1] https://suricata-ids.org/
[2] https://www.inlab.de/balance.html
[3] http://www.projectclearwater.org/

telecommunication implementations rely on well defined protocols, such as Session Instantiation Protocol (SIP), and a standardized architecture with components to initiate, control, and manage calls. IMS functions consist of software packages implemented in VMs or Docker containers as depicted in Figure 3.7. In the case of Docker (MERKEL, 2014) implementation, it uses an embedded DNS in the Docker host.

Figure 3.7: Base scenario 2 - IMS internal components with parallel chaining through replicas.




Source: the Author

Each of the internal components of the the current Clearwater IMS implementation are described below:

- **Bono** - Bono components are the Proxy-Call Session Control Function (P-CSCF) standard interface to IMS clients, and they serve as edge proxies providing access points for the client's connection to the Clearwater system. Multiple instances of Bono are replicated, and load balancing is performed by DNS requests. Although a client is anchored to a specific Bono instance during its registration, it can move to another node if connection fails.
- **Sprout** - This component implements the Serving Call State Control Function (S-CSCF) and Interrogating Call State Control Function (I-CSCF) interfaces of the IMS standard, and it provides a scalable SIP register and authoritative routing proxy. In summary, they are responsible for handling client authentication and the IP multimedia Service Control (ISC) interface to application servers.
- **Homer** - Homer is an XML Document Management System (XDMS) responsible for storing settings for each user of the system in XML documents. Documents are created, read, updated, and deleted using a standard XML Configuration Access Protocol (XCAP) interface.
- **Vellum** - Vellum is the component responsible for maintaining long-lived state in the deployment. It includes multiple distributed storage clusters: *Cassandra* for storing authentication credentials and profile information, *etcd* for sharing internal Vellum configurations, *Chronos* for reliable time service, and *Memcached* for storing registration and

session states.

- **Astaire** - Astaire components help provide elasticity to Clearwater. When nodes experience scaling up/down actions, Astaire is responsible for synchronizing data across all nodes that use *Memcached* as a database. The Astaire node runs as a daemon in background, acting when triggered by a "reload" request. When requested, it pulls data from the *Memcached* servers that have data that must live locally and push them to the local *Memcached* server.

For the sake of fairness of results in comparison, we use the same workload as NFV-VITAL (CAO et al., 2015). To simulate scaling up actions, in addition to the previous standard PC workstation, we run a Clearwater Docker implementation[4] on top of a COTS server with 64 CPU cores and 64 GB of memory running an Ubuntu server 16.04, Linux kernel 4.4. By doing this, we can understand the effects of memory and CPU capacities over the analyzed metrics. Also, akin to the security chain case (Section 3.3.2.1), we used the `tc qdisc` tool to provision two levels of network capacity (*i.e.*, 10 Mbps and 50 Mbps) between IMS internal components to insert bottlenecks. The HSS database is populated with 50k subscribers and the IMS system is subjected to an increasing demand of REGISTER requests (ranging from $2^5$ to $2^{12}$), generated by `sipp`[5]. Each request flow is as follows: a client makes a non-authenticated REGISTER message; then, it receives a 401 – authentication fail response from the server; next, the client sends the REGISTER message with valid authentication parameters and waits for a 200 – OK response from the server. Because of the lack of isolation between containers, it is worth mentioning that using a container-based implementation of VNFs hampers the individual accounting of three metrics: CPU steal usage, committed memory, and cache miss rate.

### 3.3.2 Representativeness of monitoring metrics

Given the scenarios presented in Section 3.2.2, we now assess to what extent the monitoring metrics resultant from Section 3.2.1 can quantify performance impacts on service chains. Thus, we discuss results of the experimental evaluation in both base scenarios.

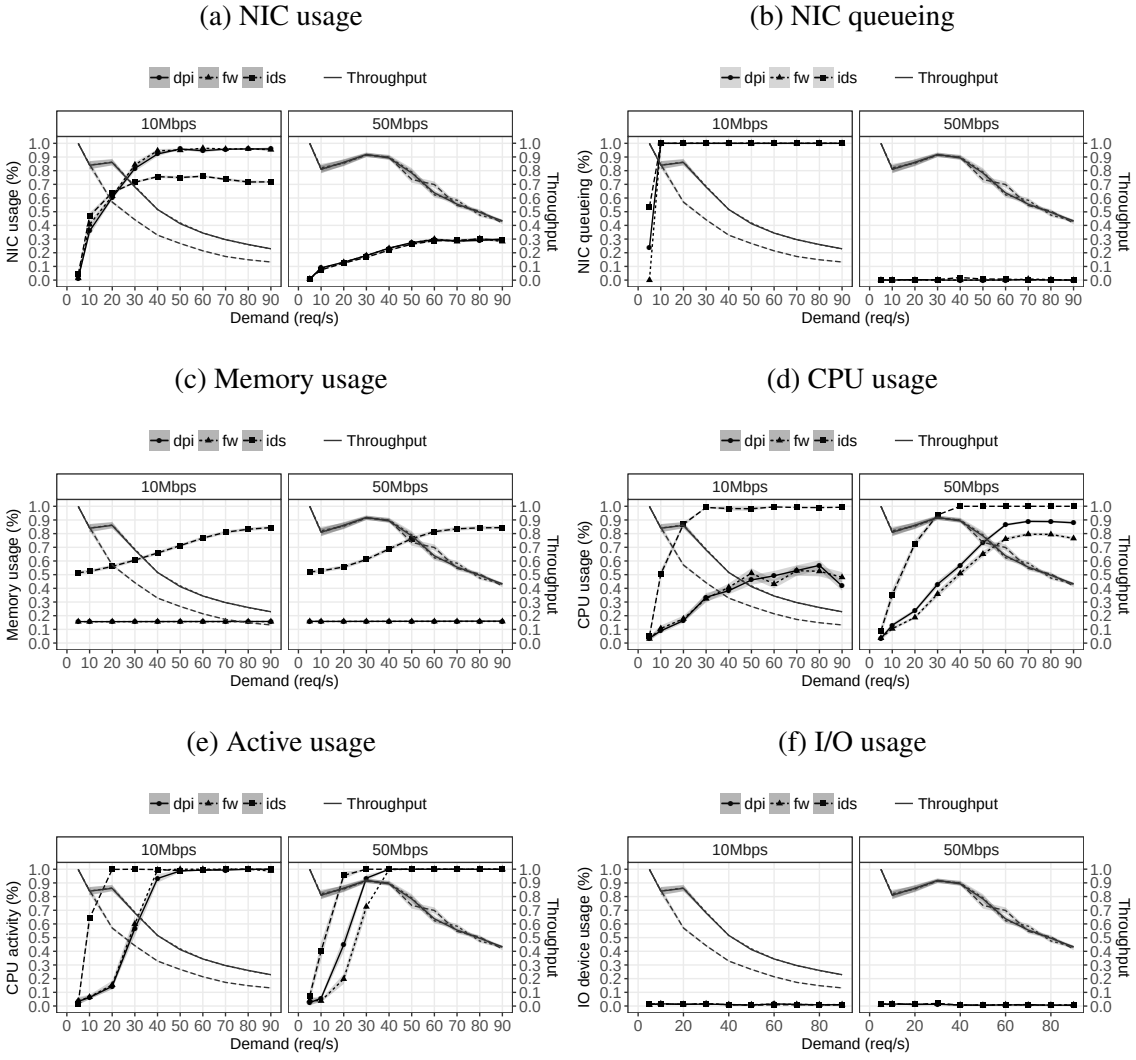
#### *3.3.2.1 Capacity constraints in a security chain*

Figure 3.8 depicts the measurements of monitoring metrics from the three chained VNFs. Each line/point in the charts of Figure 3.8 depicts the average of 30 repetitions and the computed standard error with 95% confidence level. We expect this experiment to reflect a major impact from the FW and IDS, since the FW is the first VNF in the chain, and the IDS has a deeper level of inspection. It is also expected that throughput increases from 10 Mbps to 50 Mbps network capacity. This means that the effect of network restriction in throughput occurs with a higher

[4]https://github.com/Metaswitch/clearwater-docker
[5]http://sipp.sourceforge.net/

demand of requests with 50 Mbps than with 10 Mbps. The gray lines depict the application throughput measured when distinct VNFs are bottlenecks, where the gray dashed line refers to the throughput obtained when the restrictions are imposed to the IDS. As depicted in the results, the end-service performance behaves as expected: when provisioned capacity is of 10 Mbps the application throughput starts to fall down at a demand of 20 requests per second, while with 50 Mbps the throughput decrease starts at 40 reqs/s. It is important to highlight that bottlenecks imposed to the IDS have a greater impact then when imposed to the other two VNFs.

Figure 3.8: Monitoring metrics in a security chain with capacity constraints




Source: the Author

Regarding the extent of metrics representing application performance, NIC usage (Figure 3.8a), NIC queueing (Figure 3.8b), and CPU usage (Figure 3.8d) are the best candidates for characterizing the performance of the security chain. When network capacity is restricted, the three VNFs exhibit NIC queuing, thus being a good signal of bandwidth congestion. As

we add more capacity to each NIC, the queuing on NIC interfaces almost ends and NIC usage reduces. However, the application throughput still experiences a drop even with increased network capacity, indicating that even a small amount of NIC queueing can signalize performance issues. The behavior of CPU usage also helps to explain the throughput decrease: when the provisioned network capacity is 50 Mbps, the VNFs' CPU usage increases alongside the demand, and when the IDS reaches around 90% the fall on application throughput starts. On the flip side, memory usage, active usage, and I/O usage do not aid in the performance diagnosis. As the demand of requests increases, the IDS is the only VNF that exhibits an increasing memory usage, going from 50% to around 90%. This behavior is the same with 10 Mbps, and 50 Mbps provisioned capacities and does not reflect the application throughput performance. In the case of CPU activity and I/O usage, their behavior remains indifferent to network capacity, and the measurements do not express the variances of application throughput.

Despite the fact that these three metrics (*i.e.*, NIC usage, CPU usage, and NIC queueing) are the most useful for diagnosing application performance, the challenge of establishing thresholds for them remains open. For example, in the case of NIC usage, should one consider amounts of resources occupation superior to 50% as a warning signal? One can consider this as a small value for capacity planning purposes. Regarding CPU usage, is 80% acceptable for signalizing degradations? Such questions raise doubts about using threshold-based approaches, as in the works of Pfitscher et al. (2016), Gember et al. (2013), and Cao et al. (2015), for bottleneck identification purposes. We further delve in this threshold establishment discussion in Chapter 4, Section 4.3.

Finally, it is important to discuss the importance of service-related metrics. Service rates, service times, and response time are prominent to reflect end-to-end performance, but they depend on detailed knowledge of request types. As discussed in the state-of-the-art section, the works of Gallardo, Baynat and Begin (2016), Suksomboon et al. (2016), and Prados-Garzon et al. (2016b) argue that by applying analytical models one can relate service arrival rates (*i.e.*, the demand of requests in our experiments) to the throughput of each VNF to compute service rates and service times. With an assumption of homogeneous requests, and considering Little's Law, it is easy to determine a VNF service time as the ratio between demand and throughput. Also, by summing the service times of all VNFs in a chain, it is possible to compute the response time of the entire chain. However, as we relax the homogeneity of requests assumption, and consider that each VNF will process heterogeneous flows, the difficulty to model the service time for each particular request type increases. We chose not to explicitly plot these metrics because they are redundant to throughput measurements.

#### *3.3.2.2 Stress injections in security chains*

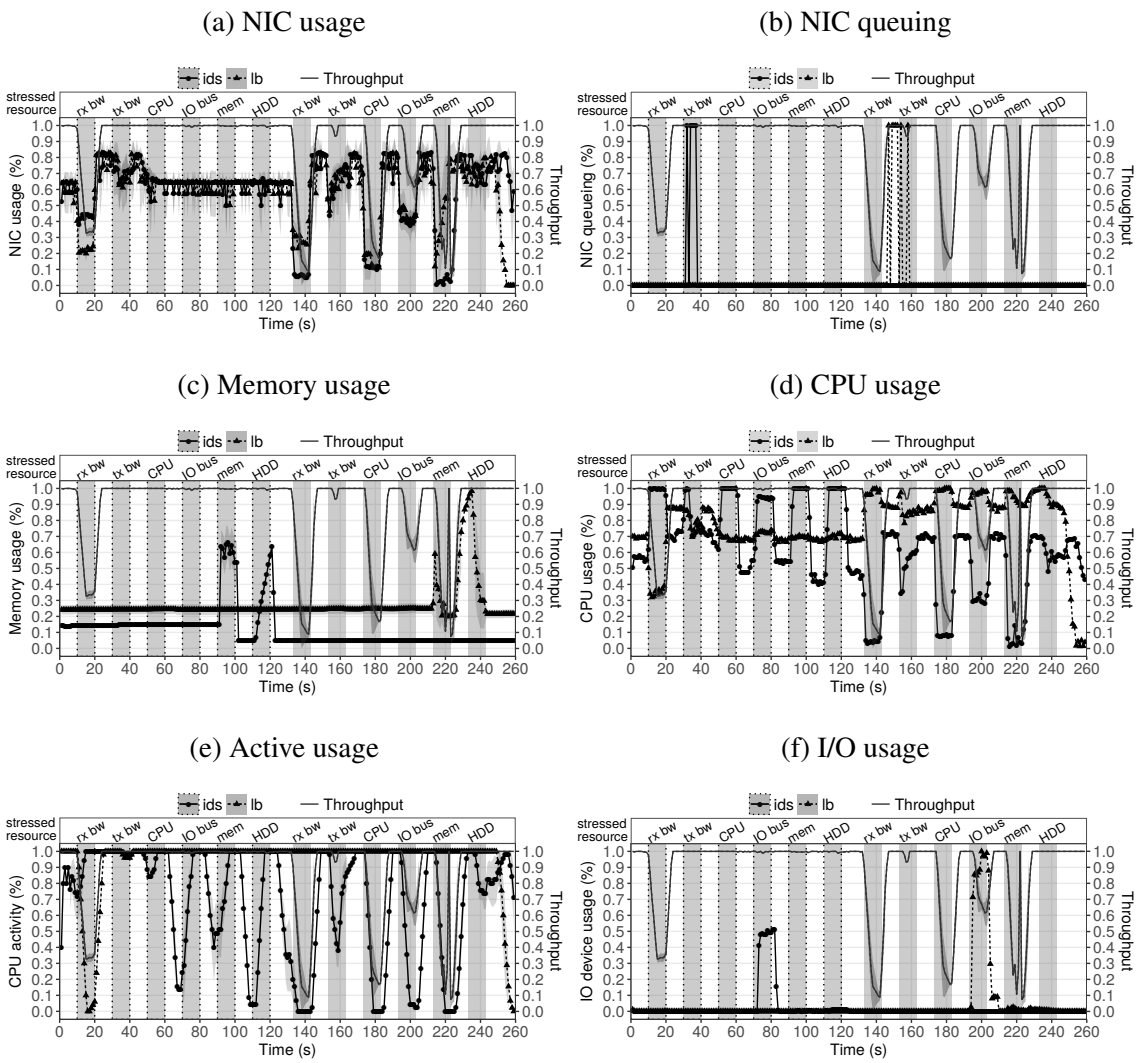
In addition to bottlenecks generated through capacity constraints, we analyze how metrics behave when security chains are subjected to stress injections. Figure 3.9 depicts the behavior

of the metrics measured in such scenario. The intention of injecting stresses is to impact in application throughput independently of the VNF being stressed. However, results show that the IDS becomes a bottleneck only when it is subjected to an inbound flood traffic (rx bw), causing degradation on application throughput. In the opposite, the injection of a network output flood (tx bw) in the LB is the only type of stress that does not cause a performance degradation to the Web service. At a first glance, one can argue for the observation of NIC usage (Figure 3.9a) as a signal of performance problems: the lower the NIC usage, the higher the degradations in throughput. This claim is valid if we consider the service as a whole, but, if the aim is to point out bottlenecks, the metric fails. By looking more closely to results, what we see is that NIC usage points out the wrong VNF: during the interval 10-20 the LB is the VNF with the lower NIC usage, when, indeed, the stressed VNF is the IDS. Such behavior repeats for all the other intervals where application throughput degrades.

Since the observation of NIC usage may imply in misleading results, we must now evaluate how the remaining metrics behave against the injections. NIC queuing (Figure 3.9b) occurred in both VNFs when we submit an output flood traffic load. The TCP congestion algorithm explains such behavior; when the flood starts, TCP tries to increase the congestion window size, but, as the available bandwidth has not changed, packets remain in backlog until the congestion window reduces to the previous value, which is proportional to the bandwidth. Regarding degradations, Figure 3.9b shows that the output flow traffic does not influence any of the VNFs, being thus the NIC queuing irrelevant for bottleneck diagnosis in the stress injection scenario. Similarly to NIC queuing, memory usage (Figure 3.9c) is useless for diagnosing bottlenecks, stress loads in memory, and HDD makes an increase of memory usage, which does not alter the application throughput. In turn, CPU usage (Figure 3.9d) is most prominent for diagnosing degradations; for all the cases where application throughput struggled, the CPU exhibits a high load in the stressed VNF. However, the metric also presents several false positives: there are cases where CPU is high and there are no performance reduction. An insight appears from the CPU activity (Figure 3.9e): when the LB is subjected to any kind of load, the activity of the IDS reduces. Unfortunately, this insight is not effective for diagnosing purposes, because it is exclusive for this VNF and scenario. Finally, I/O usage (Figure 3.9f) is valuable for one of the five performance issues: when injection occurs in the I/O bus of IDS, the I/O usage increases and accurately signalizes the performance issue.

In summary, after the analysis of stress injections in the security chain scenario, we can conclude that both CPU activity and I/O usage metrics must be included in the set of representative metrics. Thus, considering these metrics inclusion, the resultant set we establish so far consists of *NIC usage, NIC queueing, CPU usage, CPU activity, and I/O usage*. However, following the assumption that for diagnosing heterogeneous VNFs require the observation of different metrics, we assessed metrics behavior in the Clearwater IMS scenario. Section 3.3.2.3 discusses the results of this assessments.

Figure 3.9: Monitoring metrics in Security with stress injections



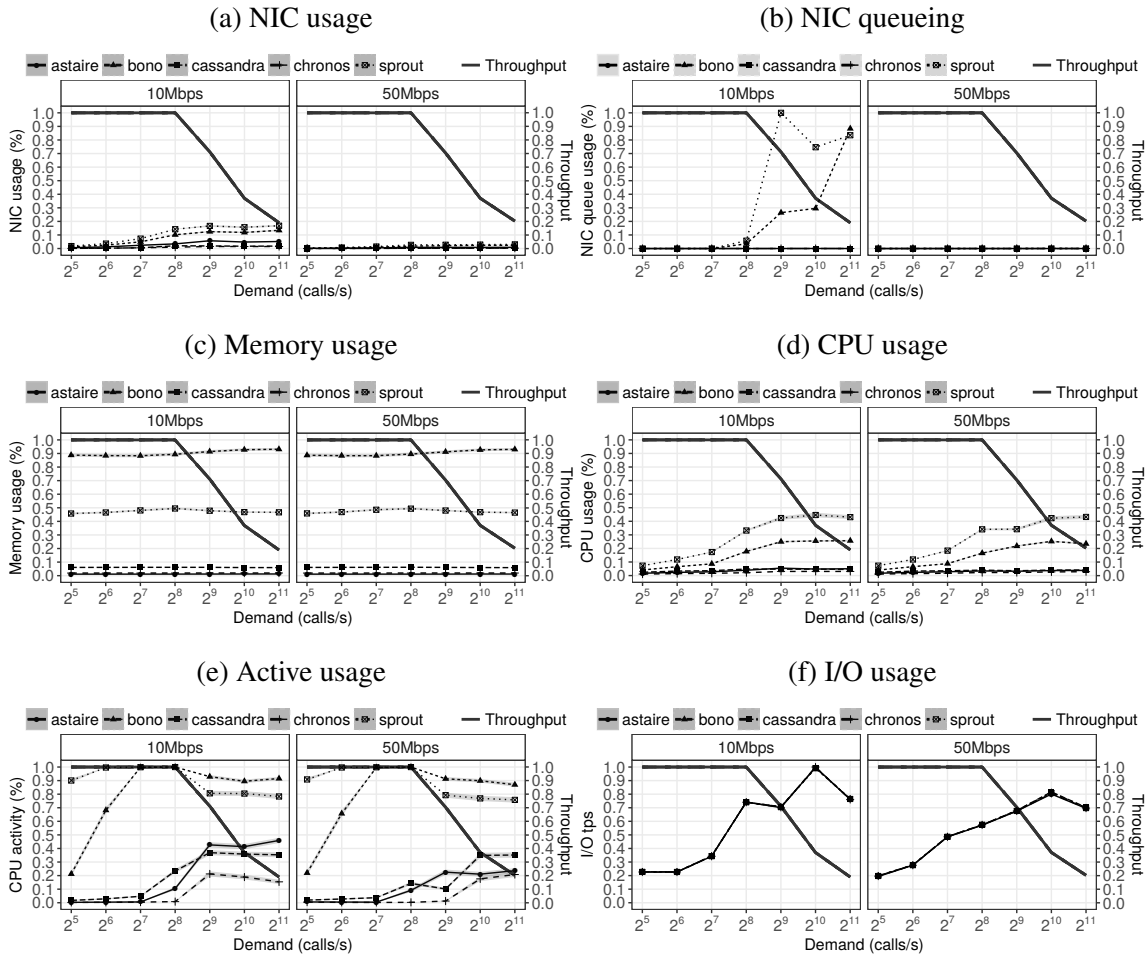

Source: the Author

### *3.3.2.3 Capacity constraints in Clearwater IMS*

In this section, we evaluate the representative results of monitoring metrics in the second base scenario. Figure 3.10 depicts the measurements of the metrics' assessment of Clearwater IMS running on top of a PC workstation. To have an objective comparison between scenarios, we also plot the application throughput as end-service metric. The results show that network capacity does not influence the application throughput, and that Clearwater is able to sustain up to $2^8$ calls/s. As we increase the available bandwidth from 10 Mbps to 50 Mbps, the call throughput maintains exactly the same levels. With a demand of $2^{11}$ calls/s, Clearwater throughput is only able to cope with around 20% of submitted requests, signaling a severe drop in performance. Since the workload to which we submitted the scenario is composed solely of REGISTER re-

quests, Sprout and Bono nodes are the more likely to impact on service performance, given that Bono is the entry point for clients and Sprout deals with the authentication aspects of connections. Hence, both VNFs have a much higher probability of becoming bottlenecks in the service chain.

Figure 3.10: Monitoring metrics in Clearwater IMS with capacity constraints




Source: the Author

Four metrics aid in diagnosing the drop in application throughput in this scenario: memory usage, CPU usage, CPU activity, and I/O usage. Figure 3.10c shows that the Bono node consumes around 90% of available memory when subjected to a small demand ($2^5$), and that Sprout nodes use around 40% of available memory. Considering that memory consumption grows with the increasing demand, and that the sum is greater than 100%, we can conclude that performance degradations occur due to memory swapping in the server. In the case of CPU usage, from a demand of $2^8$ calls/s and on-wards, there is an increase of both Sprout and Bono nodes consumption. However, even when combining the consumption of both nodes ($0.45 + 0.25 = 0.7$), we are far away from the resource usage limit, which complicates the task of establishing mon-

itoring thresholds. CPU activity is also meaningful, as the value increases proportionally to demand. Finally, I/O usage directly relates to the degradation of the application throughput, as the throughput falls drastically when I/O devices are used more. Still, it is worth mentioning that measurements of I/O transactions per second were taken in the host, and thus, it is difficult to identify which VNF is the cause of performance degradation. Nonetheless, one can argue for NIC queueing observation, since it indicates network capacity is restricted. However, network capacity is clearly not the root cause of the throughput drop because bandwidth improvement does not solve the problem.

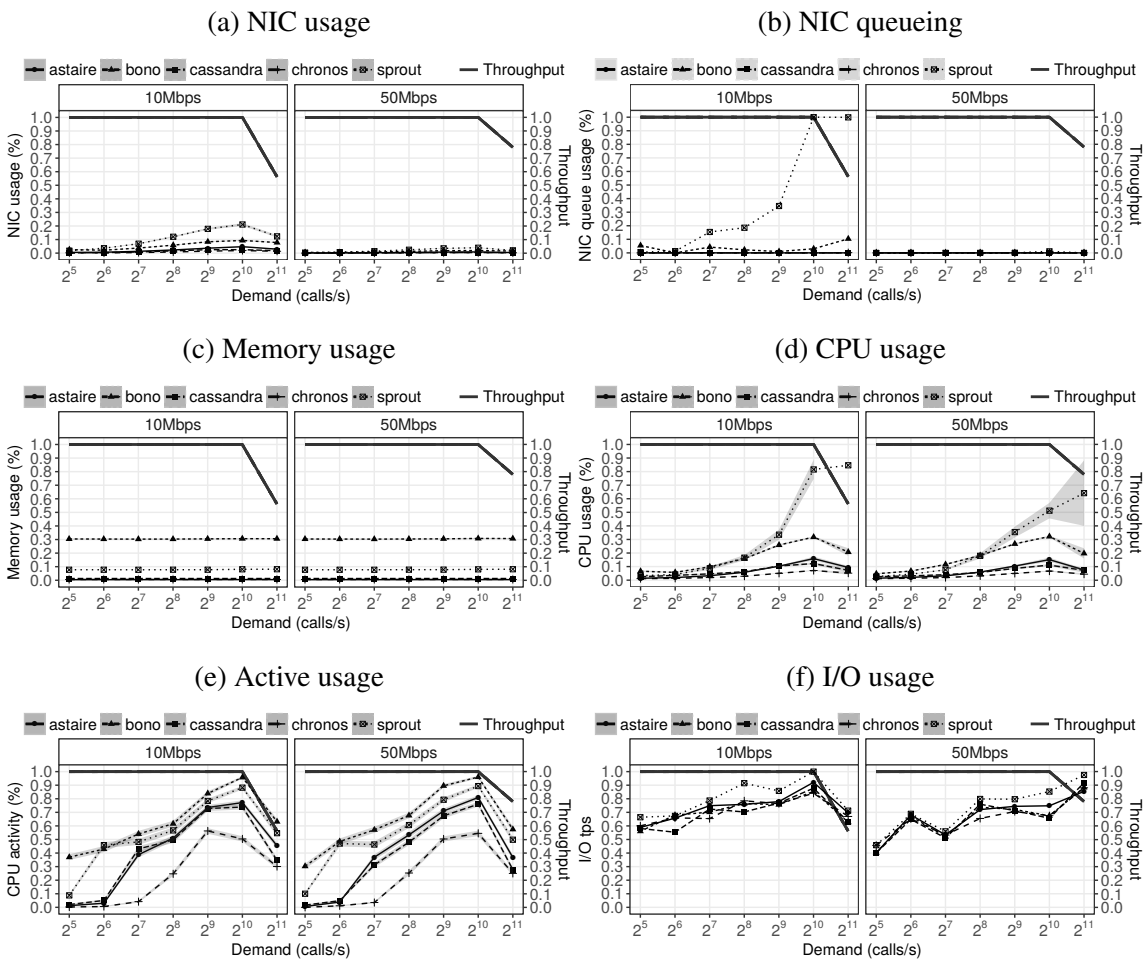
To understand the effects of scaling up resources, we run the Clearwater scenario in a hardware with more resource capacity. Figure 3.11 depicts the measurement results for this experiment. As the plots show, the amount of sustained demand increases with the scaling up action; throughput application only starts to fall with a demand of $2^{10}$ calls/s. Other two aspects must be highlighted: first, the NIC capacity influences the general performance; second, the sum of memory usage maintains constant and does not exceeds available capacity. For example, independently of the requests demand, the sum of each VNF memory usage is around 50%. The remaining of metrics exhibit a behavior similar to the case without scaling, which allows us to conclude that, because of its behavior in the scenario before vertical scaling, the memory usage must be included in the set of representative metrics: *NIC usage, NIC queueing, CPU usage, CPU activity, I/O usage, and memory usage*.

The difficulty in establishing thresholds is confirmed in the Clearwater IMS scenario. Even though we can define a set of representative metrics, values of metrics depend on available hardware capacity. For example, while the CPU usage indicates saturation of VNFs with around 40% in the PC workstation, in the COTS server, this values is far away from the amount where performance degradation starts (80%). The same occurs with memory usage, which is useful only in the tests with the PC workstation scenario. Such discrepancy suggests that the analysis of metrics must be done case by case, and they must be combined for better results.

## 3.4 Chapter summary and remarks

In this chapter, we addressed the first research question (RQ-1). After surveying the NFV evaluation scenarios from the literature, in Section 3.2.1 we studied what metrics have been used to analyze NFV performance, and converged our research to fourteen performance metrics divided in four categories: *network-related*, *memory-related*, *processor-related*, and *service-related*. Then, we established two base scenarios that reflect the the most commonly evaluated NFV scenarios: one with sequential security VNFs and another with parallel IMS internal components. In Section 3.2.2 we found that NFV chains are either sequential or parallel, and that *security* VNFs are the most used specific purpose functions. Also, we found that the Clearwater Project implementation of IMS internal components allows parallel chains. Finally, we performed experimental evaluations in three scenarios derived from the base ones: security

Figure 3.11: Monitoring metrics in Clearwater IMS after vertical scaling




Source: the Author

chain with capacity constraints, security chain with stress injections, and clearwater IMS with capacity constraints (in this case with and without hardware scaling).

From the analysis, we conclude that *NIC usage, NIC queueing, CPU usage, CPU activity, I/O usage, and memory usage* are the metrics with the most potential for identifying bottlenecks. However, given that these metrics behave distinctly from one scenario to another, we envisage that, for precise results, they must be combined, and their importance must be calibrated in each particular case. Given such considerations, Chapter 4 will continue the research by addressing the second and the third research questions, which objectives are, respectively, to find out how to combine metrics in a model, and how to adjust the model according the particularities of each scenario.

# 4 MODELING SERVICE CHAIN PERFORMANCE

Seeking to understand the extent of traditional metrics for bottleneck detections in NFV service chains, Chapter 3 reviewed the literature and provided results from experimental evaluations. Such results have shown that the relevance of metrics varies significantly according to the evaluated scenario. In this Chapter, we address the second research question, which investigates how to combine the identified set of relevant metrics in a model able to quantify performance degradations in general service chain scenarios. To do so, we combine the results from empirical observations with concepts from queueing networks theory to develop a generalized mathematical model.

As this model results in a numerical representation of performance degradation caused by individuals VNFs, we call it `guiltiness`. We argue that each VNF is responsible for an amount of performance degradation. Thus, the more a VNF degrades in comparison to the others in the chain, the higher the `guiltiness` of the VNF. In the following sections, we discuss how to model such value as a metric and how to adjust the metric to environment particularities.

## 4.1 Modeling the guiltiness of VNFs

The assessments of Chapter 3 show that, from the universe of monitoring metrics used in NFV literature, the subset $M$ of representative metrics contains CPU, memory, NIC, and I/O usages, as also NIC queuing and CPU activity. Another fact the evaluation results reveal is that each metric has a different level of importance on signaling performance degradations, which brings us to the second research question:

**RQ-2:** *Having found a set of performance representative metrics, how can they be combined in a model to accurately quantify performance degradations caused by individual VNFs in general NFV scenarios?*

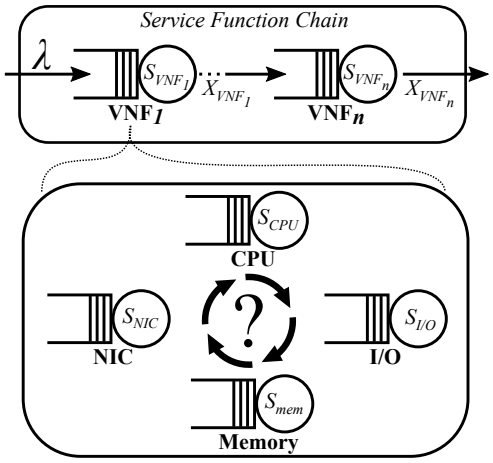
To answer RQ-2, we must first find a proper method to relate these metrics and gauge the different impacts each one has on the SFC performance. As proposed by Koizumi and Fujiwaka (2015), one approach to quantify the performance $P_v$ of a VNF $v$ in a service chain, is to combine each metric $m_i$ in the representative set $M$ through a simple weighted sum:

$$P_v = \sum_{i}^{m} w_i \times m_i \tag{4.1}$$

where, weights $w_1 ... w_m$ define the importance of each metric, and $m$ is the number of metrics in $M$. Although this approach can provide an estimate of performance, it depends on linear regressions based on historical measurements, which can hinder its adoption for real-time bottleneck detection. As a first step towards circumventing such limitation, we study the queueing networks theory to model the `guiltiness` of performance degradation caused by each VNF

on a service chain. Figure 4.1 depicts an abstracted view of such model. Let us consider each VNF as a queue, which receives requests with an arrival rate $\lambda$, process and forwards them to the next queue (*i.e.*, VNF) in the chain with an outgoing rate $X_v$ (also known as throughput). In such model, the amount of time a request stays in a queue $v$ is denoted as residence time $R_v$, and the total time taken by a request to transverse the entire service chain denotes the chain's response time $Rt_c$, which is the sum of all residence times. Also, as we uncovered in Chapter 3, four resources can relate to the internal processing of a given VNF: CPU, NIC, memory, and I/O devices; we discuss the internal relations of these resources later in this chapter, but for now, consider they are uncertain.

Figure 4.1: Queueing networks model for SFC




Source: the Author

One of the most well-known tools for modeling residence times in queueing networks is the Utilization Law (GUNTHER, 2007). The Utilization Law defines the residence time $R$ of a request in a queue as the relation between service time $S$ and queue usage $\rho$, with $\rho$ impacting $R$ in an exponential manner. This means that at low values, $\rho$ has a small impact on the performance, but as $\rho$ increases the performance falls quickly. This happens because, at high utilization, the resource cannot cope with the processing demand, thereby leading to the queuing of processing tasks. Considering that four queues represent the internal components of a VNF, represented by the four resource usage metrics (*i.e.*, CPU, NIC, memory, and I/O usages), and that request residence times in such resources depend on their utilization, one can model the residence time of a request in a VNF $v$ as:

$$R_v = \frac{S_{cpu}}{1 - \rho_{cpu}} + \frac{S_{NIC}}{1 - \rho_{NIC}} + \frac{S_{mem}}{1 - \rho_{mem}} + \frac{S_{I/O}}{1 - \rho_{I/O}} \tag{4.2}$$

By observing Equation 4.2, it is clear that, to be precise, the model depends on measurements of each queue usage as well as of the correct estimation of service time. In the former,

monitoring tools can provide online values about queues. In the latter, to find out correct values of $S$ requires a deep knowledge of how each queue performs the processing of requests. Another option is to rely on historical observations of arrival rate and throughput to estimate service time. However, such dependency of historical data returns us to the same problem we had by using the simple weighted sum, as it hinders the detection of performance issues in real-time.

The heterogeneity of VNFs in NFV scenarios also increases the complexity of performance modeling and leads to uncertainty. In queueing networks models, to have accurate estimates, analysts must understand the relations between the internal components of each modeled node; requiring knowledge about the order that requests visit components and the time they consume in each one. NFV scenarios can include VNFs of distinct types, commonly provided by different vendors, and thus, an accurate modeling of VNFs performance requires an in-depth inspection of their behavior. The results from Section 3.3.2 also supports this uncertain relation among internal components: while the performance of the security VNFs is more sensitive to CPU and NIC capacities, the performance of IMS components is characterized by having multiple influences.

We deal with the aforementioned problems (*i.e.*, uncertainty about relations between queues, and knowledge of queue internals) in a twofold approach. First, as in the weighted sum models, we add a weight for each resource as an approximation of service time. Second, considering our finding of representative metrics, we include the resource activity ($A$) to gauge the importance of each resource. As we mentioned early, the study of Wang, Zhao and Zheng (2005) proposed the active duration time as a metric to detect bottlenecks. The original meaning of active duration time corresponds to the period of time that a node is in a non-idle state. However, as each resource can show fluctuations of low consumption, we use a threshold to establish when the resource is idle. $A$ is calculated in the following way: a monitor collects $P$ samples of resource usage ($\rho$) and for each value greater than a threshold $T$ it increments a counter $C$. Afterwards, $A$ is computed as the ratio between $C$ and $P$. The threshold $T$ is set by network operators according to their perspective of relevant load. As a start point, $T$ can be the measured value of $\rho$ when there are no requests being sent to the VNF.

By including weights and resource activities we can now start to model the `guiltiness` metric. As we adapted the Utilization Law to our purposes, we define `guiltiness` of a VNF $v$, $G_v$, as an approximation of the normalized value of residence time $R_v$. This means that, the maximum value of $G_v$ will be $1.0$, which corresponds to the maximum observation of residence time possible[1]. By using weights and resource activity, we can now define the `guiltiness` $G_v$, for any VNF $v$, on the performance degradation of a service chain as:

$$G_v = \sum_{m \in M} w_m . A_m . \frac{1}{1 - \rho_m} \qquad , M = \{CPU, mem, NIC, I/O\} \tag{4.3}$$

[1] In fact, the residence time of a queue can grow to the infinite. We assume this only occurs in critical situations, and neglect this in the model.

In addition to the relevant resources for distinct VNFs, as modeled in Equation 4.3, we argue for the measurement of queueing in the NIC interface for a more precise signal of performance degradations. The results from the assessment of metrics' representativeness in Sec. 3.3.2 show that NIC queueing is a valuable signal of performance degradation, since the occurrence of network queuing indicates that a VNF is not able to cope with the incoming demand. In one of our papers (PFITSCHER et al., 2018), we also recognized the need for a term to contemplate NIC queuing occurrence on the valuations of performance degradations. Such claim is also supported by the findings of Wu, He and Akella (2015), which argues for the instrumentation of every buffer/queue of VNFs' internal components. Thus, we propose a fifth term that relates CPU activity to average usage of NIC queue:

$$G_v = \sum_{m \in M} w_m . A_m . \frac{1}{1 - \rho_m} + w_5 . \frac{(A_{CPU} \times \rho_{NIC_{queue}} - A_{CPU})}{1 + \rho_{NIC_{queue}}} \\ where, M = \{CPU, mem, NIC, I/O\} \tag{4.4}$$

The fifth term of $G_v$ corresponds to the active time taken by the CPU to process NIC queued packets. Considering the assessment results (Section 3.3.2) showed CPU activity tend to exhibit high values when the VNF is submitted to any load level, our intent is to grow CPU activity impact on `guiltiness` proportionally to NIC queueing. In our first `guiltiness` proposal (PFITSCHER et al., 2018) we settled the following relation between CPU activity and NIC queuing:

$$G_v = ... + w_2 \times A_{CPU} + ... - w_4 \times \frac{A_{CPU}}{1 + NICqueue} \tag{4.5}$$

where, $w_2$ and $w_4$ define the relevance of these terms. The idea is that if queuing does not occurs in NIC, the amount that $A_{CPU}$ will impact on `guiltiness` reduces relatively to the difference between $w_2$ and $w_4$. For instance, assume $w_2 = w_4 = 0.2$ and observe the results provided for the first security scenario (Figure 3.8). In such, CPU activity grows to $1.0$ for all the nodes, independently of provisioned network capacity. However, NIC queuing only occurs when network capacity is restricted to 10Mbps, using 100% of buffer size (2500 Bytes). By replacing the last values in Equation 4.5, $G_v$ results in $0.2 \times 1.0 - 0.2 \times \frac{1.0}{1+2500} \approx 0.2$, which means that the impact of $A$ is fully maintained on the VNFs' `guiltiness`. For the case where queuing does not occurs (50 Mbps network), $G_v$ is $0.0$, which implies that CPU activity is not impacting on `guiltiness`. Hence, the fifth term added to the model (Equation 4.4), in this Thesis, emerges naturally from evaluating the Equation 4.5 that describes the first `guiltiness` model (PFITSCHER et al., 2018).

## 4.2 Adjusting weights for each scenario

In the previous section we answer RQ-2 by providing a model able to combine the $M$ representative metrics. However, as discussed in the representativeness analysis, the importance of each metric varies severally from one scenario to other, which requires tuning the `guiltiness`'s parameters. Thus, in this section, we investigate how the model can be adapted for scenarios particularities by adjusting the model's weights. This investigation applies to answer the last research question:

**RQ-3:** *Considering that weights of the model should be adjusted to scenarios' peculiarities, how to adjust the diagnostic model according to varying setups in scenarios?*

Weights in Equation 4.4 have two purposes: normalize values and gauge metrics. Because the aim of `guiltiness` is to produce a value to denote how much a VNF is responsible for degrading the performance of a service chain, the domain of $G_v$ is $\{G_v | 0 \leq G_v \leq 1\}$, where $G_v = 1.0$ means that the likelihood of $v$ being cause of degradations is 100%. As a consequence, we must establish weights that normalize the sum of terms to be at max $1.0$. For the sake of explanation, let us neglect the fifth term and assume the other four are equivalent (with same weights). Considering each term produces a value that comes from the ratio between the resource activity ($A$) and its usage ($\rho$), and that both $A$ and $\rho$ are subjected to the same range of values than $G_v$, the default values of weights $w_1$ to $w_4$ are equal to $0.0025$. With this value, when a given resource is fully used (*i.e.*, $\rho$ and $A$ equal to 100%) the relative term will be in charge of $0.25$ of `guiltiness`. Regarding the fifth term, the value of $w_5$ express the percentage that NIC queuing impacts on `guiltiness`, using $w_5 = 0.1$ implies that NIC queuing accounts for 10% of performance degradations. Thus, with this value, in a case without NIC queuing the maximum value of $G_v$ is $0.9$.

In addition to normalization, the importance of each resource in general performance may vary because of individual aspects of service chains and network functions. For example, if the CPU has a higher impact on service chain performance than the memory, $w_{CPU}$ must be greater than $w_{mem}$. The results presented in Section 3.3.2 also support this claim for adjustments: while NIC usage, NIC queuing, and CPU usage are better signals in the security scenario with capacity restrictions, memory usage and I/O usage to diagnose performance issues in the Clearwater IMS. To the end of differing metrics' impact according to scenarios, we must find a method to properly adjust the equation weights.

Let us again assume that a VNF service chain is a queueing network model. As such, we can adapt queueing network laws to our purpose. First, recall that the residence time is the time that a given VNF takes to process a request; then, according to the *General Response Time Law*, the sum of the residence time of each node in the chain is the total chain residence time (GUNTHER, 2007). Therefore, if we take into account that $G_v$ is a representation of residence time in VNF $v$, the sum $G_c$ of all $G_v$ values in a service chain $c$ is the total residence time, *i.e.*, response time. With this assumption in mind, we can isolate the equation weights to fine

tune their values. For a better understanding of the derivation, we first split the five terms of `guiltiness` in Equation 4.6:

$$
\begin{aligned}
t_1 &= w_1.A_{CPU}.\frac{1}{1-\rho_{CPU}}, \qquad & t_2 &= w_2.A_{mem}.\frac{1}{1-\rho_{mem}} \\
t_3 &= w_3.A_{NIC}.\frac{1}{1-\rho_{NIC}}, \qquad & t_4 &= w_4.A_{I/O}.\frac{1}{1-\rho_{I/O}}, \\
t_5 &= w_5.\frac{(A_{CPU} \times \rho_{NIC_{queue}} - A_{CPU})}{1+\rho_{NIC_{queue}}}
\end{aligned}
\tag{4.6}
$$

Then, in Equation 4.7, we derive $G_c$ to extract weights. Note that $w_1$, $w_2$, $w_3$, $w_4$, and $w_5$ are constant for all $v$ in $c$. Bare in mind this assumption implies in a generalization of resources' importance for all VNFs in a given service chain.

$$
\begin{aligned}
G_c &= \sum_{v \in c}^{n} G_v \\
&= \sum_{v \in c}^{n} (w_1\, t_1 + w_2\, t_2 + w_3\, t_3 + w_4\, t_4 + w_5\, t_5) \\
&= \sum_{v \in c}^{n} w_1\, t_1 + \sum_{v \in c}^{n} w_2\, t_2 + \sum_{v \in c}^{n} w_3\, t_3 + \sum_{v \in c}^{n} w_4\, t_4 + \sum_{v \in c}^{n} w_5\, t_5 \\
&= w_1 \sum_{v \in c}^{n} t_1 + w_2 \sum_{v \in c}^{n} t_2 + w_3 \sum_{v \in c}^{n} t_3 + w_4 \sum_{v \in c}^{n} t_4 + w_5 \sum_{v \in c}^{n} t_5
\end{aligned}
\tag{4.7}
$$

where $n$ is the number of VNFs in the chain $c$.

Equation 4.7 has the following outcome: to fit $G_c$ as a representation of the chain response time $Rt_c$, we must find a quintuple $<w_1, w_2, w_3, w_4, w_5>$ that makes the relation exact to the metric observations. To achieve that, the first step is to aggregate the measurements of both resource usages and activities and then compute the sum of terms from all VNFs. Next, each of the five terms in Equation 4.7, as well as, the respective $<Rt_c>$ are stored in a historical information database, with which, we can compute the weights that make $G_c \cong Rt_c$.

We rely on a hybrid learning procedure, combining both neural networks and nonlinear regression, for such weights computation. The reason why we combine these methods is to avoid fluctuations of regression results. Considering that not all scenarios are available *a priori*, and that non-linear regressions take into account the entire dataset of measurements, outliers can directly affect weight estimation. As we filter the training dataset, neural networks perform more abstract regressions, which makes them less sensitive to outliers. On the flip side, the neural networks depend on a large dataset to result in precise predictions, hindering the utilization of such method solely. Thus, in the procedure, the training data consists of metrics (*i.e.*, $t_{1,\ldots,5}$, $Rt$) and weights (*i.e.*, $w_{1,\ldots,5}$). Training can be either manual (the operator inserts training data) or automatic (using measurements and regressions). Once there is enough information to perform

the predictions, the hybrid learning procedure combines results from neural network and regressions to compute the weights. Algorithm 1 presents the learning procedure using automatic training.

**Algoritmo 1** Hybrid learning algorithm.

1: $nn \leftarrow$ NEURALNETWORK()
2: $dataset_{nn} \leftarrow \emptyset$
3: $dataset_{nlr} \leftarrow \emptyset$
4: $w^{default}_{1,...,5} \leftarrow [0.0025, 0.0025, 0.0025, 0.0025, 0.1]$
5: $w^{current}_{1,...,5} \leftarrow w^{default}_{1,...,5}$
6: **procedure** LEARNWEIGHTS($t_{1,...,5}, Rt$)
7: $Rt_{Norm} \leftarrow Rt/Rt_{Max}$
8: $dataset_{nlr} \leftarrow dataset_{nlr} \cup (t_{1,...,5}, Rt_{Norm})$
9: $w^{nlr}_{1,...,5} \leftarrow$ NONLINEARREGRESSION($dataset_{nlr}$)
10: $w^{nn}_{1,...,5} \leftarrow nn$.PREDICT($t_{1,...,5}, Rt_{Norm}$)
11: **for** $x \leftarrow (current, default, nn, nlr)$ **do**
12: $G^x_c \leftarrow$ GUILTINESS($w^x_{1,...,5}, t_{1,...,5}$)
13: $r^2_x \leftarrow$ RSQUARED($G^x_c, Rt_{Norm}$)
14: **end for**
15: **if** $r^2_{nlr} >$ *threshold* **then**
16: $dataset_{nn} \leftarrow dataset_{nn} \cup ([t_{1,...,5}, Rt_{Norm}], [w^{nlr}_{1,...,5}])$
17: **end if**
18: **if** $r^2_{nn} >$ *threshold* **then**
19: $dataset_{nn} \leftarrow dataset_{nn} \cup ([t_{1,...,5}, Rt_{Norm}], [w^{nn}_{1,...,5}])$
20: **end if**
21: $nn$.TRAIN($dataset_{nn}$)
22: $r^2_{max} \leftarrow$ MAX($r^2_{current}, r^2_{default}, r^2_{nn}, r^2_{nlr}$)
23: $w^{current}_{1,...,5} \leftarrow$ WEIGHTS($r^2_{max}$)
24: UPDATEMONITORS($w^{current}_{1,...,5}$)
25: **end procedure**

For each new set of metric measurements the algorithm decides between weights estimated through regressions and neural networks. For doing so, Algorithm 1 leverages two training datasets: $dataset_{nlr}$ that contains all measured data, and $dataset_{nn}$ that contains only the weights that achieve the desired accuracy. Algorithm 1 starts when the metric measurements of a service chain (*i.e.*, $t_{1,...,5}, Rt_{Norm}$) are retrieved. Next, in lines 7 and 8, it computes the normalized response time $Rt_{Norm}$ and save the values into the general training dataset ($dataset_{nlr}$). Then, if it has enough stored information, the algorithm calls NONLINEARREGRESSION and PREDICT (lines 9 and 10) to compute the set of weights that fit the `guiltiness` function with the metric history, using the nonlinear regression and the neural network, respectively. After the weights are computed, the algorithm checks `guiltiness` accuracy. To do so, it computes, in lines 11 to 14, the coefficient of determination (R-squared) of the fit using `guiltiness` values from each weight set (current, default, neural network, and nonlinear regression). If the computed R-squared values are greater than a threshold (lines 15 to 20), the knowledge base of the neural network ($dataset_{nn}$) is updated to include a new match between the mea-

sured values and the computed weights ($[t_{1,...,5}, RtNorm]$, $[w_{1,...,5}]$). Next, the algorithm calls TRAIN , so the neural network can learn with the filtered measurements. Finally, the algorithm calls UPDATEMONITORS to renew the weights used by each VNF monitoring agent to compute `guiltiness`.

Although we propose an adaptive algorithm to define the model weights, it depends on the capacity of operators to monitor end-metrics. We argue that the combination of default weights $w_1 = w_2 = w_3 = w_4 = 0.0025, w_5 = 0.1$ with the activity of resources can represent the majority of scenarios (see results from Section 4.3). In cases in which learning is involved, we recommend feeding the knowledge database with data from both understressed and overloaded chains. By doing so, the mechanism can learn weights based on the intermediate cases.

## 4.3 Guiltiness evaluation

To assess the ability of the model to identify bottlenecks and estimate end-service performance degradations, we performed experimental evaluations in the two base scenarios (*i.e.*, security and IMS VNFs). First, in Section 4.3.1, we analyze how the model applies to detect performance degradations using default weights for terms. Next, in Section 4.3.2 we show the benefits of adjusting weights for each scenario. To that end, we run the learning procedure in each of the four evaluated scenarios (*i.e.*, security chain with capacity constraints, security chain with stress injections, and Clearwater IMS with and without scaling) and plot adjusted `guiltiness` values. It is worth highlighting that we use the same environment setups than the previous experiments.

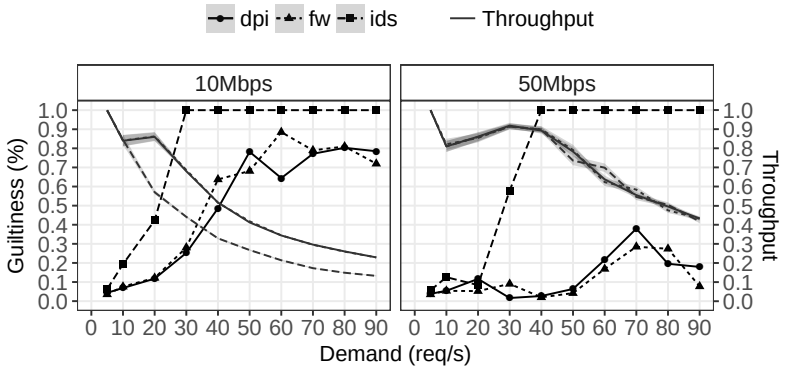
### 4.3.1 `Guiltiness` results using default weights

As we previously discussed in Section 4.2, when the importance of the `guiltiness` metrics to performance degradation is balanced, the default weights ($w_1$ to $w_4$) for terms $t_1$ to $t_4$ are equal to $0.0025$. Also, using $w_5 = 0.1$ implies that NIC queuing accounts for 10% of degradations. We now assess how `guiltiness` explains the application throughput of the evaluated scenarios when these weights apply.

Figure 4.2 depicts `guiltiness` results for the security chain with capacity restrictions scenario. In both provisioned capacities `guiltiness` unequivocally identifies the performance degradations. With 10 Mbps network capacity, the metric increases to a higher level as soon as application throughput starts to fall, pointing out the IDS with 100% of `guiltiness` from 30 req/s onward. Also, the firewall and the DPI exhibit a much smaller `guiltiness` when compared to the IDS, which is expected given the higher throughput achieved when these VNFs are bottlenecks. The 50 Mbps case shows a clear distinction among VNFs: the IDS remains as the major responsible for performance degradations, being responsible for 100% of decreases; in turn, the firewall and the DPI respond for a max of 40% of `guiltiness`,

suggesting these VNFs have a lower impact on chain performance. Although the approach of combining metrics we use in `guiltiness` allows identifying the performance issues faithfully, there is still room for improvement. By looking to the 50 Mbps case from a demand of 70 req/s and onward, both the firewall and the DPI exhibit an abnormal `guiltiness` drop, emphasizing the need for weights adjustment.

Figure 4.2: `Guiltiness` measurements on security chain with capacity restrictions using default weights

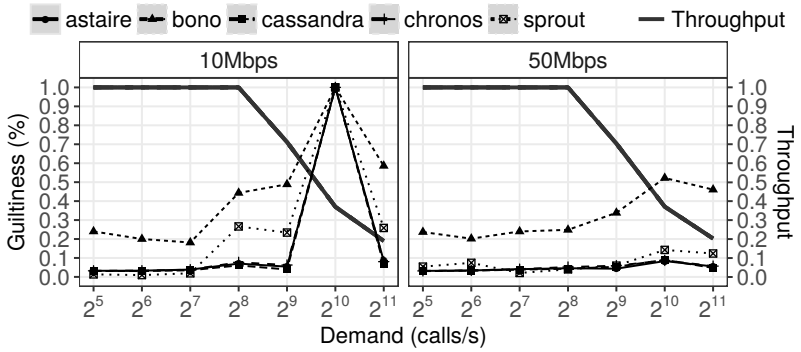



Source: the Author

Figure 4.3 depicts `guiltiness` results for the Clearwater IMS with capacity restrictions. When the available bandwidth is equal to 10 Mbps, and with a demand of $2^8$ requests per second, the `guiltiness` of Bono node increases to 45%, followed by the Sprout node (30%). The value of $G$ indicates that the system collapses with $2^{10}$ reqs/s, where all nodes have a `guiltiness` of 100%. Such behavior mainly occurs due to the high utilization of I/O and memory, and also due to the occurrence of NIC queuing (see Figure 3.10b). As we already discussed, the increase of network capacity to 50 Mpbs does not affect the application throughput, and the high level of memory and I/O usages are the only metrics that were able to explain such behavior. Regarding `guiltiness`, even by exhibiting a reduction in its general value, it indeed points out the Bono node as the root cause of problems, with a `guiltiness` of 25% at $2^8$ reqs/sec, 35% at $2^9$ reqs/sec, and 50% at $2^{10}$ reqs/sec. This behavior can be explained due to the nature of the `guiltiness` that combines multiple metrics. However, even though default weights allow to identify the performance issues, the significant reduction of `guiltiness` value in the 10 Mbps case endorses the need for model adjustment.

When we vertically scaled resources for the Clearwater IMS scenario, we observed that the degradation of throughput was postponed, occurring at a demand of $2^{10}$ requests per second. Figure 4.4 shows `guiltiness` accurately identifies such performance issue, independently of the provisioned network bandwidth. The main difference from the previous scenario is that now the node being pointed as a bottleneck is the Sprout node, instead of Bono. The higher usages of CPU and I/O in Sprout help explain this behavior. It is worth highlighting that, similarly to results from PC workstation, `guiltiness` decreases severally at a demand of $2^{11}$ req/sec with

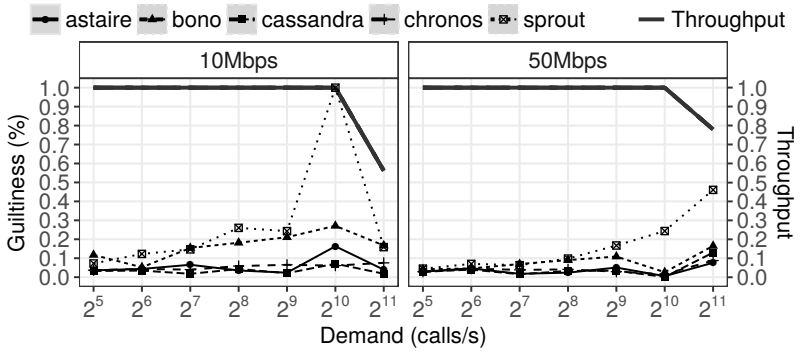
Figure 4.3: `Guiltiness` measurements on Clearwater IMS with capacity restrictions using default weights




Source: the Author

10 Mbps network bandwidth. We argue, however, that this does not hinder the metric usage for diagnosing the performance issues because it signalizes them right before their occurrence, in this case, with $2^{10}$ calls/s.

Figure 4.4: `Guiltiness` measurements on Clearwater IMS with capacity restrictions and vertical scaling, using default weights

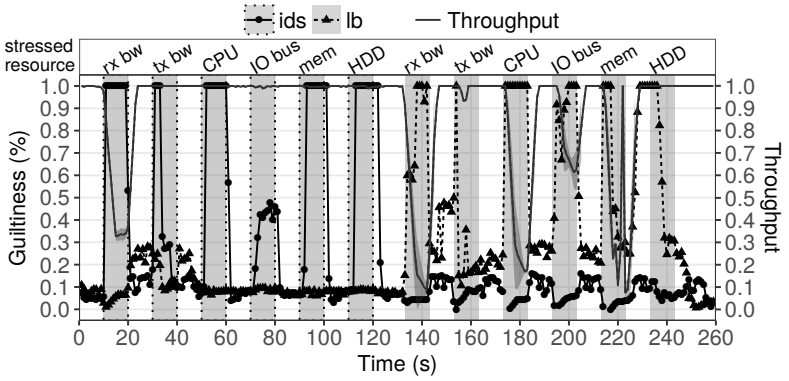



Source: the Author

Concerning the scenario with stress injections, Figure 4.5 shows that `guiltiness` can point out the correct VNF, which means, the one that is causing application throughput degradation. During the interval 0-120, the `guiltiness` value of the IDS increases in distinct moments, following the stresses it is subjected to. Similarly, from instant 140 and onwards, the VNF pointed as the cause of problems is the load balancer (which is the VNF subjected to stress loads). We argue that not providing misleadings results (indicating the wrong VNF) is the first expected result for a diagnosing metric. As such, these results show that based on `guiltiness` network operators can be sure when distinguishing each VNF impact. However, given the unexpected behavior of the IDS, in which only the RX bandwidth flood impacts on

application throughput, similarly to CPU usage (see Figure 3.9d), the `guiltiness` presents false positives; the metric wrongly points out performance issues in the intervals 30-33, 50-60, 90-100, and 100-120. On the other hand, when stresses occur in the load balancer, application throughput struggles, which is correctly indicated by the `guiltiness` value.

Figure 4.5: `Guiltiness` measurements on security chain with stress injections, using default weights



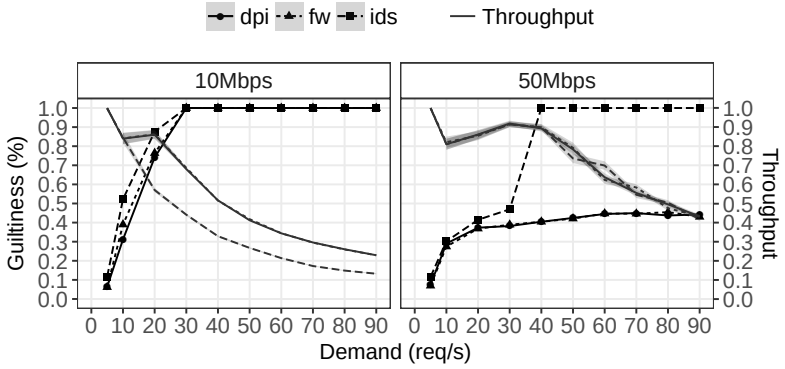

Source: the Author

### 4.3.2 `Guiltiness` results using learned weights

Given that the results from `guiltiness` with default weights leave room for improvement, we now discuss how the learning procedure applies to fill this gap through weights' adjustment in each of the evaluated scenarios. When we feed the learning procedure with data from the security chain with capacity restrictions, the procedure returns the following weight set: $(0.0006, 0.057, 0.0005, 0.0008, 0.075)$. This set of weights means that $t_5$ and $t_2$ are the most important terms of Equation 4.4 for determining the `guiltiness` of VNFs, and that $t_1$, $t_3$, and $t_4$ are almost irrelevant. The results depicted in Figure 4.6 show a clear improvement when compared to the results obtained with default weights (Figure 4.2). Using learned weights, there is a clear identification of bottlenecks: with 10 Mbps network capacity, all the VNFs exhibit high `guiltiness` values (*i.e.*, more than 80%) from 30 req/s and onward, which faithfully represents the observed application performance. Also, with 50 Mbps the learned weights reduce the abnormal `guiltiness` behavior observed in the firewall and DPI. This is explained by the lower importance given to NIC usage and CPU usage, that suffer from small oscillations with higher demands (see results of Figure 3.8), as all metrics have exponential relation to `guiltiness` when $\rho$ has high values even small decreases imply in glaring impacts.

As we go a step further, we found the best results of the learning procedure, in the Clearwater IMS scenario (Figure 4.7 and Figure 4.8). With this dataset the learning mechanism produced the following weight set: $(0.046, 0.133, -0.0001, -0.208, 0.050)$. The outcome is that $t_3$ (NIC

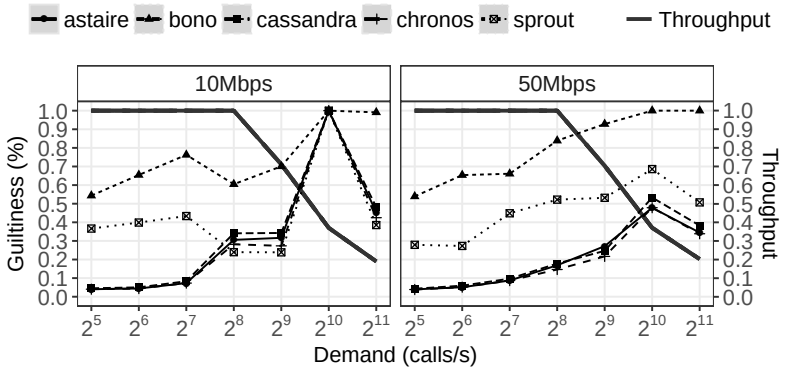
Figure 4.6: `Guiltiness` measurements on security chain with capacity restrictions using learned weights




Source: the Author

term) is almost irrelevant, $t_4$ is inversely proportional to `guiltiness` (the higher the I/O usage the lower is the guilty), and $t_2$ (memory term) is the most relevant for the `guiltiness` of VNFs. In the case without vertical scaling, `guiltiness` with the learned weights (Figure 4.7) can now provide a better characterization of the performance downfall of the 50 Mbps case, without losing the meaning for the 10 Mbps case.

Figure 4.7: `Guiltiness` measurements on Clearwater IMS with capacity restrictions using learned weights

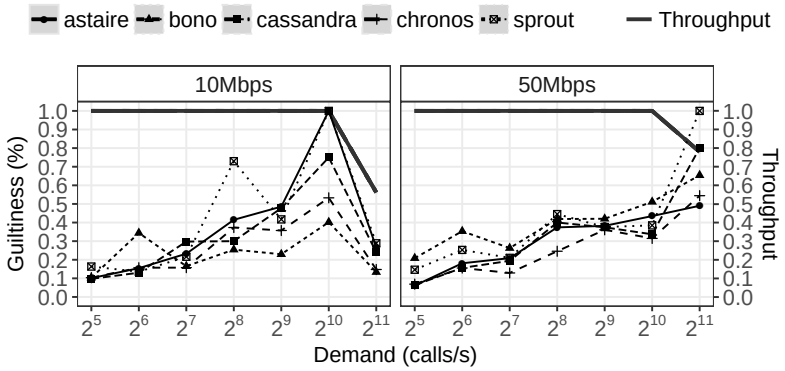



Source: the Author

The benefits of learning weights also appear in the results with vertical scaling (Figure 4.8). As we previously discussed, when the learning mechanism is not used (Figure 4.4) the Sprout node's `guiltiness` only differs from the other VNFs with $2^{10}$ reqs/s. Besides being a reliable signal of performance degradation, using the learned weights makes the `guiltiness` values be more influenced by the overall increase of metrics measurements. As a consequence, the

indication of performance issues starts earlier (with $2^7$ reqs/s), which can aid network operators to take preventive actions. The counterpart is that using learned weights in the vertical scaling Clearwater IMS scenario makes harder to distinguish the impacts of individual VNFs.

Figure 4.8: `Guiltiness` measurements on Clearwater IMS with capacity restrictions and vertical scaling, using learned weights



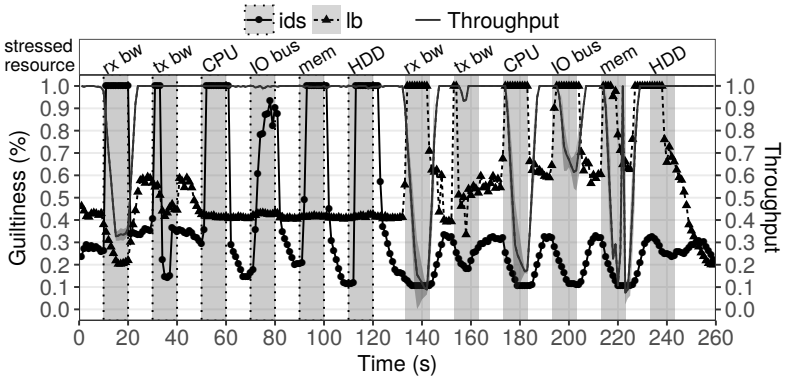

Source: the Author

Finally, we analyze the worst-case scenario for learned weights, the case of security chain with stress injection. As we already discussed, the atypical behavior of the IDS VNF makes the performance diagnosis process more challenging. This can also be justified because in Equation 4.7 we generalize weights computation by taking into account metrics measurements from the two VNFs simultaneously. Regarding the learned weights, the adaptive algorithm results in the following set: $(-0.034, -0.050, -0.0001, 0.097, -0.153)$. These values point out $t_4$ (I/O term) and $t_5$ (CPU activity *vs* NIC queuing relation) as the most relevant to `guiltiness`, and also makes $t_1$, $t_2$, and $t_3$ be inversely proportional to `guiltiness` value. The consequence, regarding `guiltiness` behavior, is that with learned weights (Figure 4.9) the values are slightly higher compared to when the model uses default weights (Figure 4.5). However, even if it does not clear the total of false positives (we discuss the absolute values in the next section), the learned weights do not affect the ability of `guiltiness` on (*i*) pointing out the VNF being the cause of performance degradation, and (*ii*) indicating situations of application throughput reduction.

### 4.3.3 Bottleneck detection precision

The analysis presented so far show that `guiltiness` can point out bottleneck VNFs and indicate when service performance starts to collapse. We now analyze `guiltiness` regarding its precision in diagnosing performance issues. Considering that in this last scenario we leveraged controlled stress injections, we can account for the number of true positive and false positives performance detection cases. For each instant of time, we assume a true positive oc-

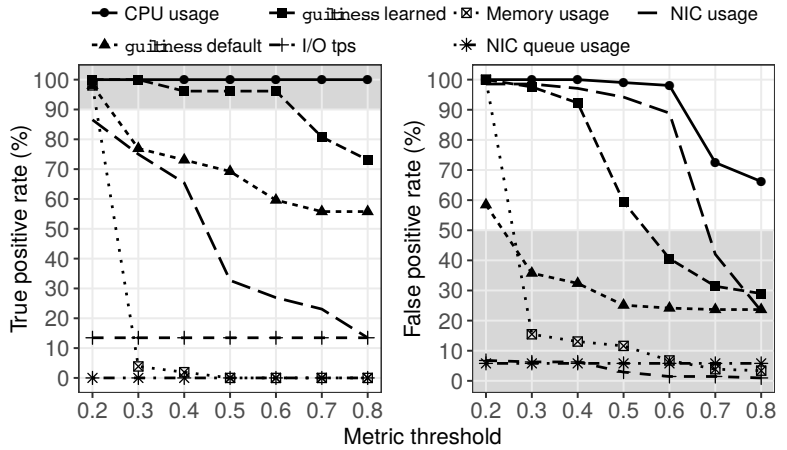
Figure 4.9: `Guiltiness` measurements on security chain with stress injections, using learned weights




Source: the Author

curs when the application throughput falls bellow 80%, and the diagnostic metric is superior to a threshold. In turn, a false positive occurs when the diagnostic metric points out a performance issue (*i.e.*, its measurement is superior to the threshold), and the application throughput remains stable (*i.e.*, throughput is superior to 80%). As metrics have different threshold values, we must find out the best case of each one for a fair comparison of performance diagnosis precision. In Figure 4.10 we plot the true positive rate (TPR) and false positive rate (FPR) for each metric using distinct values of threshold to diagnose performance issues.

Figure 4.10: True and false positive rates of metrics with distinct thresholds




Source: the Author

We argue that a good pair (metric, threshold) must prioritize the identification of performance issues over reducing the occurrence of false positives. Thus, in the precision results depicted in Figure 4.10, we highlight the cases with more then 90% of true positive rate and

less than 50% of false positive rate (in the left chart, the gray area highlights the points where metrics achieve more than 90% of TPR. In the right chart, the gray area highlights points where metrics achieve less then 50% of FPR). When the threshold is set to $0.2$, memory usage, CPU usage, `guiltiness` with default weights, and `guiltiness` with learned weights provide more than 90% of TPR. In the interval between $0.3$ and $0.6$, the set of metrics with more than 90% TPR reduces to CPU usage and `guiltiness` with learned weights. From $0.7$ and onward, CPU usage is the only metric able to detect more then 90% of the performance issues. Regarding the false positive rates, considering only the metrics that have high TPR, with a $0.2$ threshold, the `guiltiness` with default weights is the sole one to produce a FPR close to 50% (*i.e.*, $58.45\%$); then, only with $0.6$ the `guiltiness` with learned weights can result in less then 50% FPR at same time that produces high TPR. In turn, the best result for CPU usage appears at the $0.8$ threshold, with a rate of 100% for true positives and 66% for false positives. To offer a clear view of `guiltiness`' benefits we summarize in Table 4.1 the absolute values of TPR and FPR results for each metric using the best cost-benefit threshold (*i.e.*, the ones with the higher TPR and the lower FPR).

Table 4.1: Summary of true positive rate (TPR) and false positive rate (FPR) for each metric in the security chain with stress injections scenario

| **Metric** | **threshold** | **TPR** | **FPR** |
|---|---|---|---|
| I/O tps | 0.8 | 13.46% | 0.97% |
| NIC usage, as in (CAO et al., 2015), (KOIZUMI; FUJIWAKA, 2015), and (SUKSOMBOON et al., 2016) | 0.2 | 86.54% | 98.55% |
| NIC queueing, as in (WU; HE; AKELLA, 2015) | *any* | 0.00% | 5.80% |
| Memory usage, as in (CAO et al., 2015) and (KOIZUMI; FUJIWAKA, 2015) | 0.2 | 98.08% | 100.0% |
| CPU usage, as in (PFITSCHER et al., 2016), (CAO et al., 2015), (GALLARDO; BAYNAT; BEGIN, 2016), (KOIZUMI; FUJIWAKA, 2015), and (SUKSOMBOON et al., 2016) | 0.8 | 100.0% | 66.18% |
| `guiltiness` with default weights | 0.2 | 98.08% | 58.45% |
| `guiltiness` with learned weights | 0.6 | 96.15% | 40.58% |

Source: the Author

The numeric results presented in Table 4.1 confirm that `guiltiness` is able to detect performance issues, truly identifying 96.15% of performance degradations when learned weights are used. Also, in the case using default weights, `guiltiness` detects 98.08% of performance issues. However, the major benefit appears concerning false positives: while a diagnosis based on CPU usage can detect 100% of performance issues, it presents 66.18% of false positives. In turn, while `guiltiness` with default weights presents 58.45% false positives, the learned weights reduces this amount to 40.58%, a 25.6% reduction when compared to CPU usage, and 17.87% when compared to default weights.

The results from the stress injection scenario evidenced an open challenge for weights adjustment. Even though there are representative improvements regarding diagnosis precision with `guiltiness` – *i.e.*, the metric differentiates VNFs regarding their impact on performance – the anomalous behavior of the IDS VNF still produces a number of false positives. This indicates that the learning process must be refined to establish weights for each VNF,

instead of generalizing values for the entire SFC.

## 4.4 Chapter summary and remarks

In this Chapter, we addressed the second and third research questions. While RQ-2 aims to investigate how to combine the set of representative metrics in a model able to faithfully express performance degradations, RQ-3 focus on how to adjust the model according to scenarios' particularities. To answer the second research question, we make use of queueing network theory to develop a novel mathematical model combining metrics from CPU, memory, I/O, and NIC resources. We called this model `guiltiness`, which represents the amount of performance degradation each VNF is responsible for. The model consists of a weighted sum that includes resource usages, their activity, and a novel relation including NIC queuing. Results showed that `guiltiness` diagnoses the majority of performance issues, being able to identify 98% downfalls in a scenario with stress injections.

Besides the promising results of the `guiltiness` model using default weights, the number of false positives and the unexpected behavior with high demand workloads supported our claim of model adjustments. The need for adjusting the model according to the particularities of each scenario has brought us to answer the third research question. To this end, we proposed a hybrid learning mechanism that combines linear regressions with neural networks model to predict the weight set that best fit the `guiltiness` value with the normalized response time measured in an NFV service chain. Experimental results confirmed the benefits of learning weights: the cases of unexpected behaviors where mitigated, and there was a reduction of false positives (from $58\%$ with default weights to $40\%$ with learned ones). However, the proposed method for learning weights is not set in stone. In the case of security chain with stress injections, we observed that only one of the stresses (*i.e.*, NIC reception flood) had influenced the IDS VNF, and none of the diagnostic metrics was able to explain such behavior. Thus, we understand that research efforts are still required to adjust weights values, which must be subject of future work.

# 5 MONITORING BOTTLENECKS

In the previous chapters, we investigated the literature and developed a model able to both quantify performance degradations in general NFV scenarios and detect bottlenecks undoubtedly. Although proposing the state-of-the-art of bottleneck detection for NFV service chains and understanding how to adjust it according to the particularities of each scenario are central objectives of this thesis, it is also a significant tackle how to monitor NFV scenarios following specific architectural and temporal requirements.

Two characteristics of NFV introduce additional monitoring requirements: (*R1*) as discussed in Section 2.3, VNFs can be chained with other VNFs and/or Physical Network Functions (PNFs) to provide a network service (ETSI NFV ISG, 2014), and thus a feature to maintain and monitor the global service chain state is desired; and (*R2*) because of the nature of their roles, network functions require high performance packet processing (up to 10 Gbps) (MARTINS et al., 2014; HWANG; RAMAKRISHNAN; WOOD, 2015a), and thus these network functions must be monitored in near real-time (HAWILO et al., 2014; MIJUMBI et al., 2015a). Hence, in this chapter, we discuss the DReAM (Distributed Result-Aware Monitor) NFV monitoring approach (PFITSCHER et al., 2016) as a method to both fulfill these requirements and have a broad view of the `guiltiness` of service chains efficiently.

## 5.1 Monitoring strategy

Through DReAM, we address the two main requirements of NFV monitoring: (*R1*) maintenance of a global state of the network services in a scalable way, and (*R2*) minimal delay in the notification of state changes with meaningful monitoring reports. The first requirement determines that the management system needs to maintain the information about network services, composed of multiple VNFs and PNFs, without overwhelming the network infrastructure. The second requirement determines that the monitoring solution must report the state changes of NFV entities as soon as they occur, enabling fast decision-making at the NFV orchestrator. Also, the generated reports need to be sufficiently informative to allow the orchestrator to conduct a more precise VNF provisioning, resource allocation, and capacity adjustment, ideally in an autonomic loop (HAN et al., 2015). To achieve these objectives, DReAM relies on two fundamental network management techniques: *management by delegation* and *distributed monitoring*.

Commonly, the monitoring of NFV environments relies on naive or polling solutions (CLAYMAN et al., 2014; XILOURIS et al., 2014; MAINI; MANZALINI, 2014; GEMBER et al., 2013). In the former, all entities generate raw monitoring information that would be forwarded to a management system for post-processing. Afterwards, and typically offline, this raw data would be analyzed by a human or system to identify issues and make decisions (*e.g.*, capacity adjustment). This strategy, based on post-processing, delays the decision-making process. In

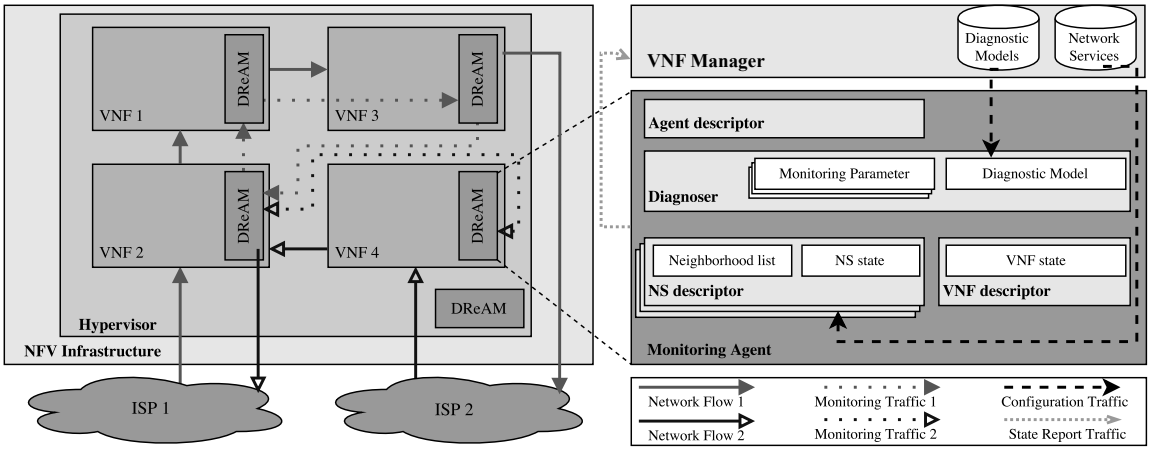
the latter, monitoring agents summarize the set of measurements for a given period, then, when requested by a manager (or orchestrator in the case of NFV), such information is sent to be used in diagnostic purposes. Besides polling can reduce the number of exchanged messages, it also delays the decision process, taking at least the polling period for an issue to be detected.

To tackle the problem of delay in decisions, DReAM proposes a *result-aware monitoring*. Result-aware monitoring is based on two concepts: management by delegation (MbD) (GOLDSZMIDT; YEMINI, 1995) and continuous distributed monitoring of state (MENG; LIU; WANG, 2011; CORMODE, 2013). In MbD, top-level managers (TLM) delegate to mid-level managers (MLM) the responsibility of executing management tasks. In comparison to traditional centralized management, MbD improves scalability by employing a number of MLMs. In DReAM, MbD allows the replacement of a single orchestrator by a set of monitors that detect, on behalf of the orchestrator, specific conditions of interest. In continuous distributed monitoring of state, each monitoring agent attaches a model to produce a diagnosis about an observed information. This diagnosis exposes the behavior of the monitored entity, which we call *result*. If the provided result is classified as an *issue* (*e.g.,* overloaded resource), such result is forwarded to the NFV orchestrator or management system (or both), to act upon it. This strategy reduces monitoring traffic between entities and orchestrator and enables near real-time decision-making.

To achieve scalable monitoring, DReAM employs a distributed monitoring strategy. In NFV environments, network functions run either inside VMs or inside physical machines. In the proposed architecture, every VM and physical machine hosts a monitoring agent. To keep track of global state information, each agent monitors both its own resources (local monitoring) and its neighbors (other VNFs or PNFs that compose the network service). On the one hand, when an agent monitors its own resources (*i.e.*, local monitoring), it collects metrics and requests to an internal diagnostic module to produce a state report (monitoring result) about the monitored VNF. On the other hand, when monitoring its neighbors, the agent periodically collects the state of each neighboring VNF; as such, if an issue is detected (*e.g.*, unreachable neighbor) an orchestrator is notified. This distributed strategy helps achieving near real-time monitoring, since it immediately exposes state changes and advertises the overall state of the network service.

To illustrate the expected monitoring agents placement, consider the scenario depicted in Figure 5.1. Each agent is responsible for monitoring VNF resources and the neighborhood that composes the network service. The continuous arrows represent the network flows, which is determined by FGs. Dotted arrows are related to neighbor monitoring, where agents request the state of their neighbors. Agents can run either inside a VM or directly in a host; this allows maintaining the state about hosts and VNFs. Suppose that the VNFs set $\{1, 2, 3\}$ composes a network service from ISP1 to ISP2, and that the respective DReAM agents are monitoring this NS. We consider that each agent is performing local monitoring, and also requesting the VNF state of other agent. With this neighborhood monitoring, DReAM fulfills the requirement of monitoring the set of entities that composes an NS.

Figure 5.1: DReAM monitoring scenario and agents architecture




Source: (PFITSCHER et al., 2016)

To determine the monitored neighbor VNF, the monitoring agent computes a hash function. The orchestrator defines a list $L$ of forwarding graphs of which an agent is a part. Then, consider that each graph has a list of identified nodes, where an $id$ represents the node's order in the graph. Thus, the function $h(id)$ returns the monitored neighbor for an agent:

$$h(id) = id + mod(t, N) + 1 \tag{5.1}$$

where $t$ is the time in minutes since midnight (*e.g.,* at 1 a.m. $t = 60$), and $N$ is the number of VNFs in the graph. The reason for expressing the time in minutes is not to be affected by clock skew on the order of a few seconds. As one can notice, the function $h(id)$ returns values between $id + 1$ and $id + N$. Thus, to diminish the chance of having results out of domain, the function $f(id)$ adjusts the outputs of $h(id)$:

$$f(id) = \begin{cases} h(id) & \textbf{if} \quad h(id) \leq N \\ h(id) - N & \textbf{if} \quad h(id) > N \end{cases} \tag{5.2}$$

for conflict treatment purposes, if the $f(id)$ returns a value equal to $id$ we simply add one to it.

As a result, the proposed hash function has two key features. First, it covers all the SFC, which means that, in a particular moment, all the VNFs monitor each other. Second, given the time dependency in the $mod(t, N)$, the function will provide a dynamic behavior, updating the monitoring structure at each time unit of $t$. We advocate that by updating the monitored neighbor, even if an agent fails, it assures that all the VNFs' states will be know at the next $t$ time unit. To exemplify a neighbor monitoring attribution, Table 5.1 simulates the function call for ten minutes and ten VNFs. The table shows that each VNF monitors a different neighbor at each time $t$.

Table 5.1: Neighborhood monitoring distribution. Each cell represents which VNF an agent will monitor

| **VNF** $id$ | time $t$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| – | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1** | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 2 | 2 |
| **2** | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 3 | 3 |
| **3** | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 4 | 4 |
| **4** | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 5 | 5 |
| **5** | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 6 | 6 |
| **6** | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 7 | 7 |
| **7** | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 8 |
| **8** | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 9 |
| **9** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 10 |
| **10** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 |

Source: (PFITSCHER et al., 2016)

## 5.2 Monitoring components

Given the focus of this Thesis is to monitor bottlenecks, we chose to present an abstracted view of DReAM components, especially regarding the diagnoser and the monitored entities. Benefits of such abstraction are twofold: first, developers can modify these modules according to their needs; second, it allows to monitor general NFV scenarios, independently of the monitoring purpose (*e.g.*, detect bottlenecks, or identifying offline VNFs). The right side of Figure 5.1 presents the general architecture of DReAM (PFITSCHER et al., 2016), the details of how it is extended to include `guiltiness`' measurements appear in Section 5.3. The nomenclature of each component in the architecture is related to the management specifications from ETSI (ETSI NFV ISG, 2014). Each component of the architecture are explained as follows.

*Monitoring agent* – This component is the primary controller of the monitoring agent module. It is responsible for providing an interface to other agents, collecting neighborhood state, updating the state of NFV services and VNFs. This concept is similar to the `vnf_dependency` element presented in the ETSI specification. Thus, each monitoring agent can be aware of the state of others. This entity has two main operations: *get_state()*, used by remote agents to request another agent's current state; and *update_diagnoser()*, used by the orchestrator to update a diagnostic model (*e.g.*, adjust thresholds, change monitored metrics, and add expected running processes in the operating system) on a set of VNFs. Each monitoring agent that receives these requests updates its diagnoser and calls its neighborhood to do the same. With this strategy, the orchestrator only needs to send an update request to one monitoring agent in a set of VNFs.

*state* – This component represents monitoring results; after the diagnosis phase, a state is de-

fined for each monitored element: the network service and the virtual network function. This data structure will store the computed state and it can receive different information values according to context. For example, a state associated to a VNF can indicate if its resources are over-provisioned, under-provisioned, or correctly provisioned. In another context, a state can indicate if an NFV service is under or over a predefined SLA threshold.

*NS descriptor* – ETSI (ETSI NFV ISG, 2014) defines a Network Service Descriptor (`nsd`) to store the information about a network service. In addition to the attributes defined by ETSI, each agent in DReAM stores its perception about the network service and a neighborhood list. The NS'es state can be defined by the monitoring agent in two ways: retrieving the neighbor's state or by applying the diagnostic model over the set of monitoring parameters.

*VNF descriptor* – As in the `nsd` definition, the ETSI defines a Virtual Network Function Descriptor (`vnfd`) to store information about VNFs. For this element, we maintain the attributes specified by ETSI incremented by the state information. The monitoring agent refreshes this state by locally monitoring and requesting the diagnoser to compute the VNF's state.

*Monitoring parameter* – In ETSI's specification, a class represents a monitoring parameter that can be tracked for each element in NFV (*i.e.*, NS, NFVI, VNF). In DReAM's architecture, this component is used to store metrics about the monitored resources. A resource can be an application, an operating system, or a physical device. For diagnostic purposes, each metric and the related measurements are used by the diagnoser to define the monitoring parameter's state. These metrics can be at different levels: application-level (*e.g.*, requests per second), OS-level (*e.g.*, resource usages), or at the level of physical resources (*e.g.*, free memory). In the case of `guiltiness` implementation (see Section 5.3), monitoring parameters must include the set of representative metrics used in the model.

*Diagnoser* – This module defines the state of an element in the monitoring agent. For this, the diagnoser implements a diagnostic model. For example, in alignment with the definition of ETSI, this diagnostic model could evaluate if a `monitoring_element` complies with a `service_deployment_flavor` requirement. A flavor determines an assurance parameter (*e.g.*, number of calls per second (cps)). A set of diagnosers can be implemented in a single monitor, where each diagnostic model can be used for a different context (*i.e.*, applications, services, or resources). It can also provide diagnostic models for resource provisioning (*e.g.*, diagnostic model for memory provisioning (PFITSCHER; PILLON; OBELHEIRO, 2014)), or a model to identify or prevent SLA violations (*e.g.*, a model that checks application requirements (GARG et al., 2014)). More specific diagnosis include *(i)* deadlock detection, where the diagnoser will compute a resource allocation graph to find cycles and, *(ii)* bottleneck detection (PFITSCHER et al., 2018), which is

aim of the guiltiness model proposed in this Thesis.

*Orchestrator and manager* – These components lead with global management decisions, as a consequence, they are considered TLMs. The former deals with resource allocation and VNF placement, the latter is responsible for the lifecycle management of VNF instances and overall coordination (ETSI NFV ISG, 2014). In the DReAM architecture exposed in Figure 5.1, the orchestrator is responsible for reacting to deployment and provisioning requests, updating diagnostic models into the manager and defining the VNFs that compose each network service. The NFV manager stores the diagnostic models and the information about the NSes. The manager uses this information to configure the neighbor list and the diagnoser in each agent. It is also responsible for receiving the states from agents and retrieve reports to the orchestrator. In the case of guiltiness, we add a learner entity to the orchestrator/manager to allow weights adjustment.

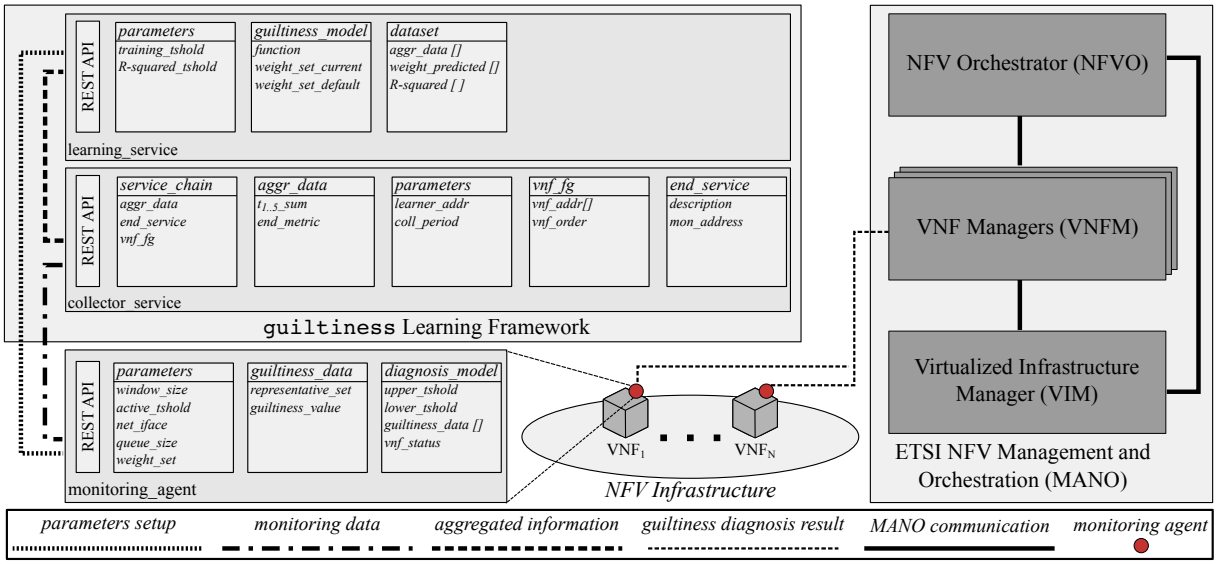
## 5.3 Implementation details of guiltiness monitoring

Although DReAM's monitoring approach serves to monitor general NFV scenarios with a large sort of monitoring objectives, the guiltiness model requires a learning process in addition to the monitoring agent that maintains the guiltiness status up-to-date in the VNF Managers. Figure 5.2 depicts the elements and their components to implement these processes, which include: (*i*) the *guiltiness Learning Framework* with two independent services, a collector and a learner; (*ii*) the *DReAM Monitoring Agent* that runs inside each VNF; and (*iii*) the ETSI NFV Management and Orchestration (MANO) reference framework. For the sake of simplicity, this figure only depicts the additional elements that are required for the guiltiness computation; those are part of the *Diagnoser* component of Figure 5.1. The remaining DReAM components are still required for the other non-bottleneck diagnosis functioning (*e.g.*, identify if a node is unreachable, maintain VNF information, and store NS data).

The learning service runs according to the sequence diagram depicted in Figure 5.3. The process starts when the *collector_service* requests the state of VNFs from the set of monitoring agents. For doing this, the collector calls *GET()* from the agents' REST API passing the string 'state'. Then, the *monitoring_agent* responds to the collector with *guiltiness_data* (includes the actual representative set of metrics and the current guiltiness value) and the computed state of the VNF. Concurrently to the VNFs' state collection, the collector also takes into account the *end_metric* value related to the end-service (*e.g.*, web server response time, VoIP jitter, and video throughput[1]). Such information is retrieved from monitoring agents running in the end-service. Next, the collector service aggregates the monitoring data (*aggr_data*) of each service chain, *i.e.*, the computed sums of the terms $t_{1,..,5}$ and the *end_metric* value of the *end_service*.

[1] In the case of diagnosis based on throughput, the values must be inverted. This inversion is required because the meaning of throughput is opposite to response time, *i.e.*, the higher is throughput the better is performance.

Figure 5.2: `guiltiness` monitoring architecture and interactions with ETSI MANO

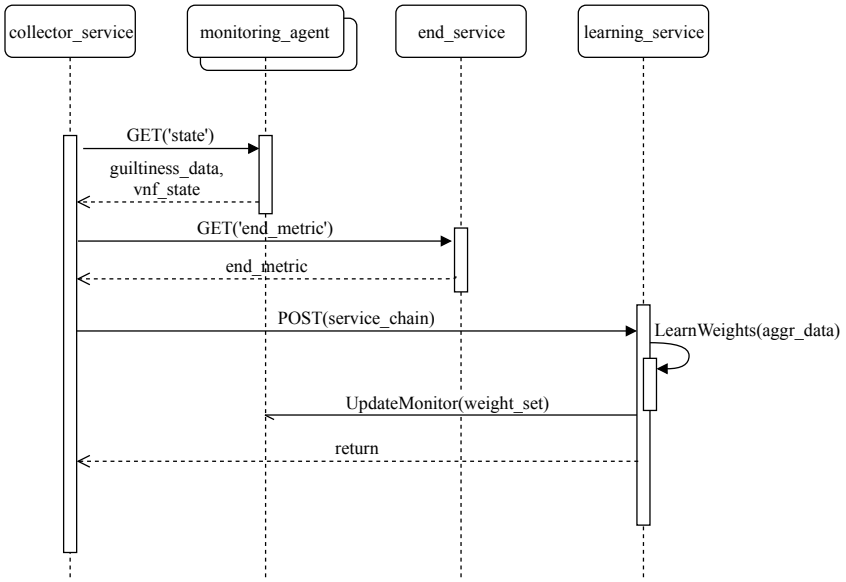



Source: (PFITSCHER et al., 2018)

After, the collector service sends the aggregated information about the *service_chain* to the learning service, through a POST call. This information includes *aggr_data*, *end_service* details, and *vnf_fg* with monitoring agent addresses. When the learning service receives the aggregated information, it first stores *aggr_data* into the service chain history (*sc_history*), to then execute the learning algorithm. For doing this, it calls *LearnWeights()* (see Algorithm 1) passing the *aggr_data* information. As far as the learning algorithm terminates and a new set of weights is defined, the learning service asynchronously calls *UpdateMonitor()* to update the weight set into each monitoring agent.

Although the process of keeping VNF states up-to-date in the VNF Managers is part of the general DReAM architecture, we decided to detail its implementation in this section because it is mandatory for bottleneck diagnosis to run properly with `guiltiness`. Figure 5.4 depicts the components interaction for VNF state change detection and neighborhood monitoring. In this process, the monitoring agent continuously (*i.e.*, at specific time intervals) updates the VNF state by calling *compute_state()*, which basically calculates the `guiltiness` value according to the measurements of *monitoring_paramenters* and process a diagnostic model. In the case of `guiltiness`, the diagnosis consists in comparing the average of measured time series (*guiltiness_data*) to two thresholds (*upper_tshold* and *lower_tshold*), and then establishing the VNF's state (*vnf_state*). We rely on our previous results (see Chapter 4, Table 4.1) to establish levels according to the risk of performance degradation: imminent, moderate, or low. The intuition behind the model is that a VNF has an imminent risk of degrading chain's performance when `guiltiness` present high values (*i.e.*, above 60%). The VNF is with a moderate risk when `guiltiness` exhibits values above 20%. When `guiltiness` measurements are be-

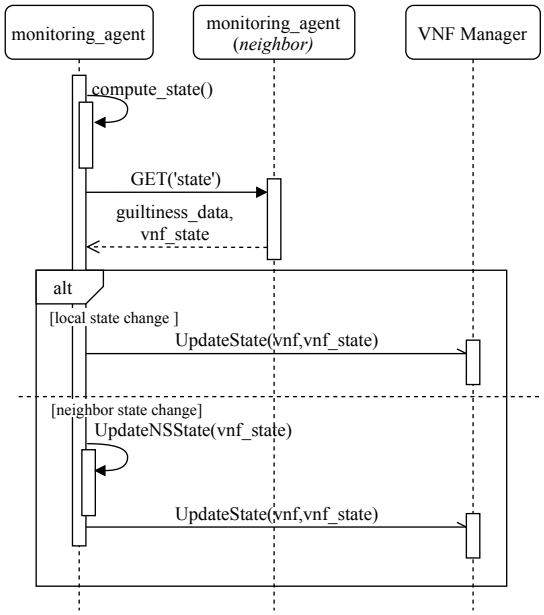
Figure 5.3: Sequence diagram for the learning process




Source: the Author

low 20% the risk of the VNF being the cause of performance issues is low. Bear in mind that a `guiltiness` of 20% means that the VNF is responsible, by itself, for 20% of overall chain performance degradation.

Figure 5.4: Sequence diagram for state change detection




Source: the Author

In addition to computing the local VNF state, the sequence diagram in Figure 5.4 depicts

the steps taken by the monitoring agent to perform neighborhood monitoring. For this, it calls *GET()* from its neighbor interface (the current neighbor is given from the *f(id)* hash function previously discussed), passing the string 'state' to indicate its interest in the neighbor's state. The neighbor monitoring agent then retrieves the *guiltiness_data* and the respective VNF state. Next, an alternative sequence runs depending on the local or neighborhood state change detection. When agents observe that a state has been changed, they inform one of the previously configured VNF Managers by asynchronously calling *UpdateState()* from the manager's API, informing in what VNF it occurred and the actual state. In the case where a state change happens in the neighborhood, the monitoring agent updates its perception about NS state by calling *UpdateNSState()* before forwarding the state change issue to the VNFM.

## 5.4 Evaluation of the monitoring approach

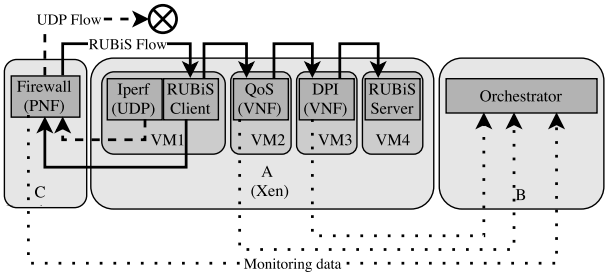
To assess whether DReAM achieves the NFV requirements, this section presents both experimental and analytical evaluations. For the experimental evaluation, we analyzed the impact on CPU utilization of the monitoring agent, the delay of state change perception by the orchestrator, and the communication overhead. For the analytical evaluation, we use linear regression to predict the CPU utilization behavior according to the number of agents. Also, we evaluated the execution complexity of obtaining a global NFV state.

### 5.4.1 Experimental evaluation

The aspects considered for monitoring approach evaluation differ from the ones that addresses the bottleneck identification model (presented in Chapter 4). Instead of assessing accuracy of bottleneck detection in distinct scenarios, the intention here is to verify whether DReAM achieves near real-time monitoring and whether it scales properly. Thus, the following evaluation results are taken from a modified version of the security chain scenario, including a PNF. The tested scenario is composed of three network functions: a firewall (PNF), a traffic shaper, and a deep packet inspector (DPI); and by two services: a TCP service and a UDP service. Figure 5.5 presents the disposition of services and functions: the firewall function blocks all UDP datagrams, the traffic shaper limits the traffic to 1 Mbps, and the DPI collects network statistics. TCP traffic is generated using RUBiS (CECCHET et al., 2003), and UDP traffic using Iperf (TIRUMALA et al., 2005). Their respective clients run in a shared virtual machine (VM1).

The environment comprises three machines: (*A*) Dell PowerEdge R620 with twelve Intel(R) Xeon(R) CPU E5-2630 v2 at 2.60GHz processors and 64 GB RAM; (*B*) Dell Optiplex 9020 with an Intel(R) Core(TM) i7-4770S CPU at 3.10GHz and 8 GB RAM; and (*C*) Raspberry Pi 2 with a quad core ARM processor at 700 MHz with 1 GB RAM. The first machine, hosting the hypervisor, runs a Debian 7 64-bit Linux with Xen 4.1; the second, the orchestrator, runs a 3.13.0 Ubuntu Linux; the third runs a Debian 3.18 Linux. All machines are connected to a

Figure 5.5: Modified version from the first base scenario




Source: (PFITSCHER et al., 2016)

100 Mbps Ethernet LAN. We used VMs running 3.2.0-4-amd64 Linux with a dedicated vCPU and 1 GB RAM. They are connected by the hypervisor's virtual link.

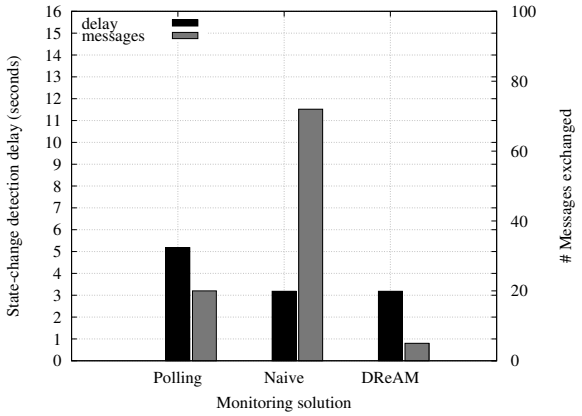
In the experiment, results are obtained from measurements of three metrics: the delay of detecting a state change, which is computed as the difference between the event occurrence and the orchestrator detection, the CPU utilization in the manager/orchestrator, and the number of exchanged bytes during all the workload run (60 seconds individually). The standard errors are related to a confidence level of 95% from an average of 30 runs. The outliers treatment excluded values with more than two standard deviations, which resulted in 1% of data rejection.

Initial assessments of the DReAM architecture concern the near real-time state changing detection requirement. Since NFV related works often rely on traditional monitoring strategies, or do not detail their monitoring solution, comparisons are made between DReAM and two common strategies: naive and polling. As we already discussed, in the naive strategy, agents forward data to the orchestrator as soon as they are measured; in the polling strategy, agents summarize the measured data in a set of values (monitoring period) and send it to the orchestrator. In addition, in both last strategies, the orchestrator is responsible for computing the diagnosis.

The experiment consists in emulating a workload in the clients, and, at random periods of time, forcing a state change in an arbitrary VNF (with a synthetic program) and capturing the event's timestamp. Moreover, when the server detects the respective VNF state change, we capture another timestamp. The difference between these timestamps is what we define as the delay for state change detection. Figure 5.6 depicts the measured delay and the averaged number of messages exchanged in the monitoring period. In this case, the confidence intervals are not significant and thus are not depicted in the plot. These results show that, in terms of delay, naive and DReAM strategies are statistically equivalent, and that the polling-based strategy is dependent on the monitoring period (which was 5 seconds in these measurements). As expected, DReAM outperforms the other approaches in terms of exchanged messages. We also measured the CPU utilization in the server. From the results, we observed that the CPU utilization in the manager was imperceptible, *i.e.*, less than 0.01% in all cases. Thus, we conclude that our

diagnostic model is simple and does not stress the server processor. Given this reason, there is no significant difference between the strategies in terms of delay in processing the VNF states.

Figure 5.6: Comparison among naive, polling and DReAM monitoring strategies




Source: (PFITSCHER et al., 2016)

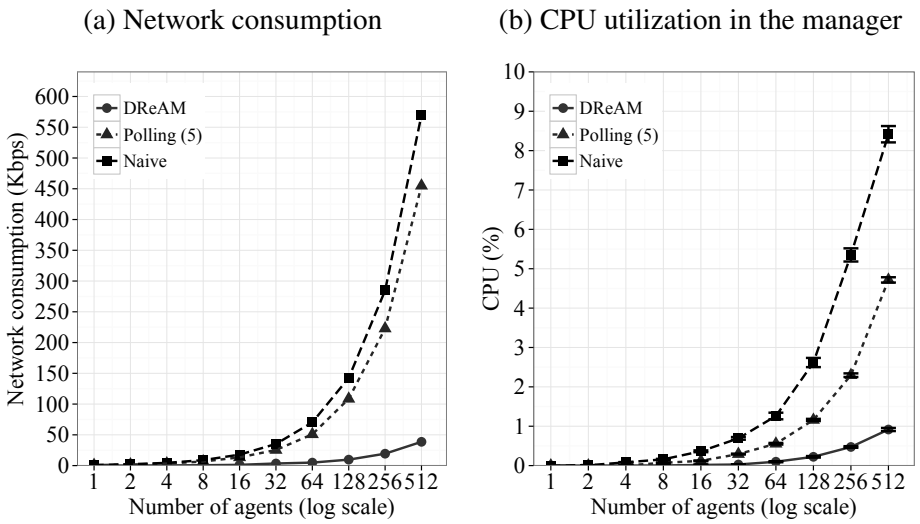
We also designed a scenario for evaluating the scalability of each strategy, where we measured the CPU utilization and the number of bytes transmitted in the network, and observed their relationship with an increasing number of agents. To provide significant results, we considered the worst case scenario for DReAM where all agents detect an event of state change simultaneously. The scalability results are plotted in Figure 5.7. As expected, DReAM has lower network overhead than the other strategies. Moreover, increasing the number of agents had a significant impact in naive and polling strategies, whereas DReAM scaled better. Despite using this simple diagnosis, all strategies support more than 512 agents without exceeding more than 10% of CPU utilization, which is negligible in terms of performance degradation.

In Figure 5.7, the x-axis is in a logarithmic scale to provide better visualization with different numbers of agents; it should be noted, however, that the relationship between the variables is actually linear (*e.g.*, double the number of agents will roughly double network traffic). Thus, we applied a linear regression model to extrapolate this behavior and determine when the DReAM strategy can provide a representative benefit for an NFV administrator. In other words, we aim to find out at what load of diagnosis DReAM would be the only solution able to meet the near real-time requirement for a certain NFV environment. Such analysis is discussed in Section 5.4.2 below.

### 5.4.2 Analytical evaluation

In the analytical analysis, we extrapolate the scalability behavior according to different values of CPU utilization diagnostic. For that, we first obtained, through a linear regression, the

Figure 5.7: Scalability analysis of the three monitoring strategies




Source: (PFITSCHER et al., 2016)

functions that provide the CPU utilization in relation to the number of monitoring agents:

$$f_{naive}(x) = 0.0178 \times x, R^2 = 0.98 \tag{5.3}$$

$$f_{polling(5)}(x) = 0.0091 \times x, R^2 = 0.99 \tag{5.4}$$

$$f_{DReAM}(x) = 0.0018 \times x, R^2 = 0.98 \tag{5.5}$$

where, $f_{strategy}(x)$ is the CPU utilization for a given strategy for $x$ number of agents, and the R-squared is the coefficient of determination. Next, we calculated the proportionality factor between the functions, considering the naive strategy as the baseline. Thus, we found that:

$$f_{polling(5)}(x) = \frac{0.0091}{0.0178} \times f_{naive}(x) \tag{5.6}$$

$$f_{DReAM}(x) = \frac{0.0018}{0.0178} \times f_{naive}(x) \tag{5.7}$$

These equations, however, do not expose a relation with the initial CPU utilization for diagnosis. Therefore, we investigated how this utilization is related to the slope of $x$. Consider that the multiplicative factor in Equations 5.3 – 5.5 are related to the sum of two variables, the CPU utilization for the diagnosis of one agent $d$, and an strategy overhead $o$. Thus, in the case of the naive strategy, we have $d + o = 0.0178\%$. By looking at results, we found that the CPU utilization for diagnosing one agent is $d = 0.00105\%$, and consequently, $o = 0.01667\%$. Thus,

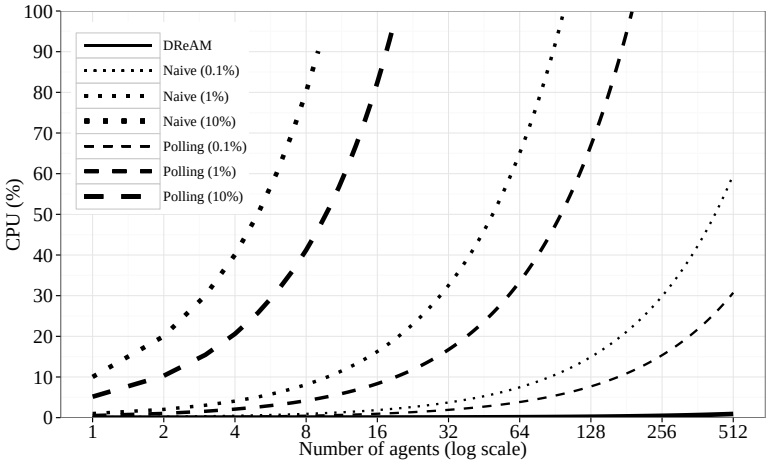
we can express the CPU utilization for diagnosing one agent with each strategy (the related R-squared value is computed in terms of the estimated values and the real observation):

$$f_{naive}(x, d) = (d + 0.01667) \times x, R^2 = 0.94 \tag{5.8}$$

$$f_{polling(5)}(x, d) = 0.5143 \times f_{naive}(x, d), R^2 = 0.98 \tag{5.9}$$

Finally, using Equations 5.5, 5.8, and 5.9 we predicted the CPU utilization in the manager for the three strategies according to the utilization of one agent diagnosis. Thus, we extrapolated for the following values of $d$: 0.1%, 1%, and 10%. Figure 5.8 presents the total CPU utilization according to the number of agents and the value of $d$. Given the distributed characteristic of DReAM, the CPU utilization for the agent does not have an impact on the server side, and so, it is not affected by the value of $d$. Therefore, we only plot one DReAM curve with the estimated values from equation 5.5.

Figure 5.8: CPU utilization prediction according to the number of agents and the CPU utilization for one agent diagnosis




Source: (PFITSCHER et al., 2016)

Results show that naive and polling strategies are directly affected by the consumption of one single agent. For example, if the diagnosis consumes 1% of CPU, the naive strategy saturates the orchestrator with more than 64 agents, and possibly affects the delay for a state change detection. In the same context, the polling strategy saturates the orchestrator CPU with less than 256 agents. With a lower consumption, *e.g.*, less than 1%, all strategies present high scalability, supporting more than 512 agents, but DReAM has the advantage of generating a smaller network overhead. In summary, DReAM is superior to the other strategies in two situations: when diagnosis processing takes more than 10% of CPU for one agent, since the naive and polling strategies only support a small number of agents (respectively 8 and 16); or when the NFV

environment is large (more than 256 agents), since the naive and polling strategies will delay the state change identification in simple diagnosis (1% of CPU utilization).

## 5.5 Chapter summary and remarks

This chapter discussed DReAM (Distributed Result-Aware Monitor), a solution for monitoring VNFs' state and detecting bottlenecks. To accomplish this with the near real-time state change detection requirement of NFV, the solution relies on two central managements concepts: MbD and distributed monitoring. The result-awareness characteristic derives from MbD, in which agents are responsible for providing a diagnosis about monitored VNF's states. At same time, the distributed monitoring allows to verify the state of all VNFs of a service chain in a scalable manner. Results from both experimental and analytical evaluations showed that DReAM is an appropriate approach for monitoring large scenarios (*i.e.*, more than 256 agents). Also, because of scalability reasons, when agents run complex diagnosis models (*i.e.*, CPU intensive analysis), result-awareness is the recommended monitoring strategy.

It should be noted that, in the result-aware strategy, the performance penalty due to monitoring and diagnosis is incurred by each VNF, and not by the orchestrator. Since it is easier to provision a single entity (the orchestrator) with extra capacity than multiple entities (every VNF), this is a potential drawback of DReAM. As such, if the diagnosis is costly and the environment is small (i.e., less than 8 agents, see Figure 5.8), the naive strategy will probably be the best choice, since it will not have an impact on VNF performance. The polling strategy takes too long detect state changes (on the order of the monitoring period), and thus fails to meet requirement *R2*. Another remark is regarding the intrusiveness of the result-aware approach. This is the weakness of DReAM, if the diagnosis is costly and the environment is small (*i.e.* less then 16 VNF) it is better for an administrator to choose the naive approach.

# 6 CONCLUSIONS

This Thesis has investigated approaches for bottleneck detection and performance quantification in NFV service chains. Although these two research topics had already been stressed in the computer networks field, NFV has additional characteristics that make these problems even more challenging. The heterogeneity of environments with all sorts of VNF purposes, the fact that VNFs are a step in the path between clients and end-services, and the unusual functioning of specific VNFs that rely on non-blocking I/O implementations, hinder the adoption of existing solutions for NFV performance quantification (*i.e.*, threshold monitoring and traditional queueing networks modeling). Also, the state-of-the-art presents no convergence when addressing performance quantification and bottleneck detection separately, and as a consequence, a plethora of metrics and approaches exist to diagnose NFV performance. By analyzing such proposals we can list a sort of reasons that prevent their adoption in general NFV scenarios: *customized metrics* are tailored for specific scenarios and require low level instrumentation; *threshold-based* and *analyticial modeling* require in-depth knowledge of NFV environments; *pattern recognition* and *statistical regressions* depend on a large set of historical measurements.

Given the exposed nuances of NFV and the current state-of-the-art limitations, we established the following research problem to steer this Thesis development:

**Problem definition:** *The problem addressed in this Thesis is the lack of an approach that can, at the same time, both detect bottlenecks in NFV service chains and quantify the likelihood of individual VNFs affecting communication performance in general-purpose NFV scenarios.*

To solve this problem, a mathematical model that relies on a weighted sum of four resource usages was introduced. By combining results from experimental evaluations, empirical observations, and analytical modeling, this Thesis pointed out that by defining coefficients for CPU, memory, NIC, and I/O usages, gauged by their respective activities, it is possible to quantify end-to-end performance impacts on general-purpose NFV service chains. In other words, such model results in a metric that estimates the `guiltiness` of a VNF being the cause of performance degradations. The work made in this Thesis results in evidence that allows answering the three research questions raised in the context of bottleneck detection and performance quantification problems. The answers provided for each question are detailed below.

**RQ-1:** *Given that the diagnosis of NFV performance has to take into account VNFs' particular characteristics, what is the set of representative metrics able to quantify the performance impacts imposed by each VNF in a service chain?*

**Answer –** Results from experimental analysis over typical scenarios with state-of-the-art metrics show that a few of them have an aptitude for diagnosing performance issues: CPU, memory, NIC and I/O usages, active percentage time, and NIC queueing. The assessments also show that metrics representativeness vary significantly from scenario to scenario, and thus, to be able to quantify performance degradations, these metrics must be combined; otherwise, they produce ambiguous results.

In this Thesis, a literature survey on NFV evaluation scenarios has been employed to establish the state-of-the-art of monitoring metrics and the base scenarios for performance assessments. The literature analysis shows that monitoring metrics are classified in four categories: *network-related*, *memory-related*, *processor-related*, *I/O-related*, and *service-related*. Also, two base-scenarios were found as representing the generality of NFV scenarios: *sequential security VNFs* and *parallel IMS internal components*. Then, experimental evaluations were employed to identify whether monitoring metrics are able to indicate end-to-end throughput degradations. Conclusions indicate that, although some metrics are representative of performance, it is hard to generalize a target threshold for them when applied to distinct VNFs. Also, each scenario has a specific set of metrics tailored to characterize its behavior, which requires a method to combine metrics and adapt to particularities.

**RQ-2:** *Having found a set of performance representative metrics, how can they be combined in a model to accurately quantify performance degradations caused by individual VNFs in general NFV scenarios?*

**Answer –** Performance quantification models can combine the set of representative metrics in a weighted sum, and specializations for each scenario are made by adjusting weights.

In this Thesis, an equation computes the amount of performance degradation imposed by each VNF. An analysis of traditional queueing network theory allowed to model requests' residence time in each VNF as a function of four resource usages (CPU, memory, NIC and I/O), their respective activities, and the queueing in network interface card. The resultant equation (referred to as `guiltiness`) consists of a weighted sum of five terms, where weights define terms' relevance according to the particularities of scenarios. The trustworthiness of the model has been shown in three scenarios fairly comparable to the ones used in the related work. The experimental results demonstrated that `guiltiness` faithfully characterizes throughput degradations in the majority of cases, the models' limitation occur in the cases where stresses do not impact on end-service performance. Such limitations suggests that weights adjustments can increase estimates precision, which requires a model fine-tunning.

**RQ-3:** *Considering that weights of the model should be adjusted to scenarios' particular characteristics, how to adjust the diagnostic model according to varying setups in scenarios?*

**Answer –** A hybrid learning mechanism that combines neural networks and nonlinear regressions allows to find out the appropriate values of weights, and as a consequence, adapt diagnosis models to environment particularities with less susceptibility to outliers than approaches that rely uniquely on regressions.

In this Thesis, historical measurements feed a hybrid learning framework to compute the equation's weights. With the assumption that `guiltiness` measures the residence time of requests in each VNF, and that the overall response time is the sum of all residence times, the `guiltiness` model was generalized to measure the performance of the entire chain. The hybrid framework relies on both nonlinear regressions and neural networks to learn the values of coefficients that approximate `guiltiness` values to normalized response times. The effec-

tiveness of the hybrid learning framework came to light in the scenario with stress injections, where `guiltiness` values with learned weights has a better trade-off between the number of false- and true-positives than default weights.

This Thesis provided arguments to support the approach used to model the `guiltiness` metric as a solution to the research problem. During the process of solving such a problem, many contributions and conclusions have been stated. First, the literature survey demonstrated that *there is no consensus on the characteristics of NFV evaluation scenarios*, with a seven-fold category of VNFs (*i.e.*, security, routing/switching, performance, RAN, EPC, IMS, and packet modification). Then, results from metrics representativeness assessments showed that *it is not possible to diagnose every performance problem using metrics from individual resources*; indeed, a model that combines metrics from CPU, memory, I/O, and NIC resources is better suited for the job. Having a set of representative metrics defined, this Thesis introduced a model to combine them to result in an estimate of performance degradations. Such model consists of a weighted sum that includes resource usages, their activity, and a novel relation including NIC queuing. Experimental results supported the conclusion that *`guiltiness` allows detecting all performance problems caused by VNF processing capacities, but may exhibit false positives when stress loads do not impact on overall performance*. To mitigate such disadvantage, this thesis introduces a hybrid learning solution to adjust the model weights according to scenarios particularities. Assessment results within such approach resulted in this Thesis' last conclusion: *the weighted sum approach is suitable to customize `guiltiness` for scenarios' particularities, but research efforts are still required to properly adjust weight values*.

Despite the fact this Thesis advances the state-of-the-art of performance quantification and bottleneck detection in NFV service chains, there are some limitations and open gaps for future research. First, it should be mentioned that it is not the focus of this Thesis to verify whether a VNFs accomplishes its design goals. For instance, if a firewall does not block UDP traffic as expected, the `guiltiness` metric cannot account for that; in turn, it can signalize if such behavior generates network forwarding performance issues. Another design limitation relates to the performance of third-party elements in the service chain. This Thesis provides a measurement of the impact caused by VNFs, and thus, the proposed metric is not able to identify performance problems in other elements, such as cables, physical switches, and physical routers. Also, even though it is possible to use `guiltiness` to diagnose VNFs hosts, the evaluations were conducted with monitors running only at VNFs' operating system level.

Finally, the approach proposed for learning weights is still not definitive. Although the hybrid learning mechanism is suitable for scenarios where VNFs in a service chain perform similarly, the cases where one specific VNF behaves significantly different than others still need appropriate learning. Thus, to find precise methods to learn such peculiarities is an open gap for future research. To this end, we envisage that learning approaches must find weights for each VNF individually, instead of generalizing values for the entire chain. Such studies on how to better gauge VNFs' anomalies are an important topic for future research.

# REFERENCES


ACM. *The ACM Digital Library*. 2018. Available from Internet: <http://dl.acm.org/>.

ADDIS, B. et al. Virtual network functions placement and routing optimization. In: *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. [S.l.: s.n.], 2015. p. 171–177.

ANDRES-MALDONADO, P. et al. Virtualized MME Design for IoT Support in 5G Systems. *Sensors*, v. 16, n. 8, 2016. ISSN 1424-8220. Available from Internet: <http://www.mdpi.com/1424-8220/16/8/1338>.

BECK, M. T.; BOTERO, J. F. Coordinated Allocation of Service Function Chains. In: *2015 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2015. p. 1–6.

BHAMARE, D. et al. A survey on service function chaining. *Journal of Network and Computer Applications*, v. 75, p. 138 – 155, 2016. ISSN 1084-8045. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1084804516301989>.

BRIM, S. W.; CARPENTER, B. E. *Middleboxes: Taxonomy and Issues*. RFC Editor, 2002. RFC 3234. (Request for Comments, 3234). Available from Internet: <https://rfc-editor.org/rfc/rfc3234.txt>.

CALLEGATI, F.; CERRONI, W.; CONTOLI, C. Virtual Networking Performance in OpenStack Platform for Network Function Virtualization. *Journal of Electrical and Computer Engineering*, v. 2016, p. 15, 2016. ISSN 20900155.

CALLEGATI, F. et al. Performance of Network Virtualization in cloud computing infrastructures: The OpenStack case. In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. [S.l.: s.n.], 2014. p. 132–137.

CAMARILLO, G.; GARCIA-MARTIN, M.-A. *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds*. [S.l.]: John Wiley & Sons, 2007.

CAO, J. et al. VNF-FG design and VNF placement for 5G mobile networks. *Science China Information Sciences*, v. 60, n. 4, p. 040302, Mar 2017. ISSN 1869-1919. Available from Internet: <https://doi.org/10.1007/s11432-016-9031-x>.

CAO, L. et al. NFV-VITAL: A framework for characterizing the performance of virtual network functions. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. [S.l.: s.n.], 2015. p. 93–99.

CAO, Z.; ABUJODA, A.; PAPADIMITRIOU, P. Distributed Data Deluge (D3): Efficient state management for virtualized Network Functions. In: *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. [S.l.: s.n.], 2016. p. 782–787.

CAU, E. et al. Efficient Exploitation of Mobile Edge Computing for Virtualized 5G in EPC Architectures. In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. [S.l.: s.n.], 2016. p. 100–109.

CECCHET, E. et al. Performance comparison of middleware architectures for generating dynamic web content. In: ENDLER, M.; SCHMIDT, D. (Ed.). *Middleware 2003*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 242–261. ISBN 978-3-540-44892-1.

CERRATO, I. et al. An efficient data exchange algorithm for chained network functions. In: *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*. [S.l.: s.n.], 2014. p. 98–105. ISSN 2325-5552.

CHEMODANOV, D. et al. A general constrained shortest path approach for virtual path embedding. In: *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. [S.l.: s.n.], 2016. p. 1–7. ISSN 1944-0375.

CHOWDHURY, K. et al. *Proxy Mobile IPv6*. RFC Editor, 2008. RFC 5213. (Request for Comments, 5213). Available from Internet: <https://rfc-editor.org/rfc/rfc5213.txt>.

CLAYMAN, S. et al. The dynamic placement of virtual network functions. In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. [S.l.: s.n.], 2014. p. 1–9. ISSN 1542-1201.

CLEARWATER. *Project Clearwater*. 2018. Available from Internet: <http://www.projectclearwater.org/>.

COHEN, R. et al. Near optimal placement of virtual network functions. In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. [S.l.: s.n.], 2015. v. 26, p. 1346–1354. ISBN 978-1-4799-8381-0. ISSN 0743166X.

CORMODE, G. The Continuous Distributed Monitoring Model. *ACM SIGMOD Record*, ACM, v. 42, n. 1, p. 5–14, 2013.

COTRONEO, D. et al. Dependability evaluation and benchmarking of Network Function Virtualization Infrastructures. In: *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. [S.l.: s.n.], 2015. p. 1–9. ISBN 9781479978991.

DATE, H. et al. Evaluation of parallel processing control of virtual switch architecture on large-scale network. In: *2014 IEEE Global Communications Conference*. [S.l.: s.n.], 2014. p. 1816–1822. ISSN 1930-529X.

DENNING, P. J.; BUZEN, J. P. The operational analysis of queueing network models. *ACM Computing Surveys (CSUR)*, ACM, v. 10, n. 3, p. 225–261, 1978.

DIEZ, L.; IZUEL, S.; AGÜERO, R. Generic Wireless Network System Modeler: Fostering the Analysis of Complex LTE Deployments. In: AGÜERO, R. et al. (Ed.). *Mobile Networks and Management*. [S.l.]: Springer International Publishing, 2017. p. 131–145. ISBN 978-3-319-52712-3.

ERAMO, V.; MIUCCI, E.; AMMAR, M. Study of Reconfiguration Cost and Energy Aware VNE Policies in Cycle-Stationary Traffic Scenarios. *IEEE Journal on Selected Areas in Communications*, v. 34, n. 5, p. 1281–1297, May 2016. ISSN 0733-8716.

ETSI, G. S. *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; IP Multimedia Subsystem (IMS)*. [S.l.], 2013.

ETSI NFV ISG. *Network Functions Virtualisation*. [S.l.], 2012. Available from Internet: <http://portal.etsi.org/NFV/NFV_White_Paper.pdf>.

ETSI NFV ISG. *Network Functions Virtualisation: Use Cases*. [S.l.], 2013. Available from Internet: <http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf>.

ETSI NFV ISG. *Network Functions Virtualisation: Management and Orchestration*. [S.l.], 2014. 184 p. Available from Internet: <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf>.

ETSI NFV ISG. *Network Functions Virtualisation: Use Cases Revision*. [S.l.], 2017. Available from Internet: <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20001v1.2.1%20-%20GR%20-%20NFV%20Use%20Cases%20revision.pdf>.

GALLARDO, G. A.; BAYNAT, B.; BEGIN, T. Performance Modeling of Virtual Switching Systems. In: *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. [S.l.: s.n.], 2016. p. 125–134. ISBN 978-1-5090-3432-1.

GALVE, J. M. et al. Reconfigurable radio access networks using multicore fibers. *IEEE Journal of Quantum Electronics*, v. 52, n. 1, 2016. ISSN 00189197.

GARG, S. K. et al. Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter. *Journal of Network and Computer Applications*, Elsevier, v. 45, p. 108–120, 2014.

GEBREMARIAM, A. A. et al. Dynamic strict fractional frequency reuse for software-defined 5G networks. In: *2016 IEEE International Conference on Communications, ICC 2016*. [S.l.: s.n.], 2016. ISBN 9781479966646.

GEMBER, A. et al. *Stratos: A Network-Aware Orchestration Layer for Middleboxes in the Cloud*. [S.l.], 2013.

GENGE, B.; GRAUR, F.; HALLER, P. Experimental assessment of network design approaches for protecting industrial control systems. *International Journal of Critical Infrastructure Protection*, v. 11, p. 24–38, 2015. Cited By 4.

GOLDSZMIDT, G.; YEMINI, Y. Distributed Management by Delegation. In: IEEE. *Distributed Computing Systems, 1995., Proceedings of the 15th International Conference on*. [S.l.], 1995. p. 333–340.

GONZALEZ, A. et al. Service availability in the NFV virtualized evolved packet core. In: *2015 IEEE Global Communications Conference, GLOBECOM 2015*. [S.l.: s.n.], 2015. ISBN 9781479959525.

GULBRANDSEN, A.; ESIBOV, D. L. *A DNS RR for specifying the location of services (DNS SRV)*. RFC Editor, 2000. RFC 2782. (Request for Comments, 2782). Available from Internet: <https://rfc-editor.org/rfc/rfc2782.txt>.

GULENKO, A. et al. Evaluating machine learning algorithms for anomaly detection in clouds. In: *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*. [S.l.: s.n.], 2016. p. 2716–2721.

GUNTHER, N. J. *Guerrilla capacity planning: a tactical approach to planning for highly scalable applications and services*. [S.l.]: Springer Science & Business Media, 2007.

HAN, B. et al. Network Function Virtualization: Challenges and Opportunities for Innovations. *Communications Magazine, IEEE*, IEEE, v. 53, n. 2, p. 90–97, 2015.

HARCHOL-BALTER, M. *Performance modeling and design of computer systems: queueing theory in action*. [S.l.]: Cambridge University Press, 2013.

HAWILO, H. et al. NFV: State of the art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC). *Network, IEEE*, IEEE, v. 28, n. 6, p. 18–26, 2014.

HWANG, J.; RAMAKRISHNAN, K.; WOOD, T. Netvm: High Performance and Flexible Networking Using Virtualization on Commodity Platforms. *Network and Service Management, IEEE Transactions on*, IEEE, v. 12, n. 1, p. 34–47, 2015.

HWANG, J.; RAMAKRISHNAN, K. K.; WOOD, T. NetVM: High performance and flexible networking using virtualization on commodity platforms. *IEEE Transactions on Network and Service Management*, v. 12, n. 1, p. 34–47, 2015. ISSN 19324537.

IEEE. *IEEE Xplore Digital Library*. 2018. Available from Internet: <http://ieeexplore.ieee.org/>.

IETF. *Domain names - implementation and specification*. RFC Editor, 1987. RFC 1035. (Request for Comments, 1035). Available from Internet: <https://rfc-editor.org/rfc/rfc1035.txt>.

JACOBS, A. S. et al. Artificial neural network model to predict affinity for virtual network functions. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2018. p. 1–9. ISSN 2374-9709.

JACOBSON, V. Congestion avoidance and control. In: ACM. *ACM SIGCOMM computer communication review*. [S.l.], 1988. v. 18, n. 4, p. 314–329.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1990.

JARRAYA, Y. et al. Multistage OCDO: Scalable Security Provisioning Optimization in SDN-Based Cloud. In: *Proceedings - 2015 IEEE 8th International Conference on Cloud Computing, CLOUD 2015*. [S.l.: s.n.], 2015. p. 572–579. ISBN 9781467372879.

JERO, S. et al. Dynamic control of real-time communication (RTC) using SDN: A case study of a 5G end-to-end service. In: *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2016. p. 895–900. ISBN 9781509002238.

JIA, W.-K. W.-K. Architectural Design of an Optimal Routed Network-based Mobility Management Function for SDN-based EPC Networks. In: *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks - Q2SWinet '15*. [S.l.: s.n.], 2015. p. 67–74. ISBN 9781450337571.

KANG, N. et al. Alpaca: Compact network policies with attribute-encoded addresses. *IEEE/ACM Transactions on Networking*, IEEE, v. 25, n. 3, p. 1846–1860, 2017.

KHEBBACHE, S.; HADJI, M.; ZEGHLACHE, D. Virtualized network functions chaining and routing algorithms. *Computer Networks*, v. 114, p. 95–110, 2017.

KIM, W. Toward network function virtualization for cognitive wireless mesh networks: a TCP case study. *EURASIP Journal on Wireless Communications and Networking*, v. 2015, n. 1, p. 1–16, 2015. ISSN 1687-1499.

KOERNER, M.; GAUL, C.; KAO, O. Evaluation of a cloud federation approach based on Software Defined Networking. In: *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*. [S.l.: s.n.], 2015. v. 2015-Decem, p. 657–664. ISBN 978-1-4673-6773-8.

KOIZUMI, S.; FUJIWAKA, M. SDN + cloud integrated control with statistical analysis and discrete event simulation. In: *2015 International Conference on Information Networking (ICOIN)*. [S.l.: s.n.], 2015. v. 2015-Janua, p. 289–294. ISBN 978-1-4799-8342-1.

KOURTIS, M. A. et al. Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network, NFV-SDN 2015*. [S.l.: s.n.], 2015. p. 74–78. ISBN 9781467368841.

KRUEGER, T. et al. Tokdoc: A self-healing web application firewall. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2010. (SAC '10), p. 1846–1853. ISBN 978-1-60558-639-7. Available from Internet: <http://doi.acm.org/10.1145/1774088.1774480>.

LAI, J.; FU, Q.; MOORS, T. Using SDN and NFV to enhance request rerouting in ISP-CDN collaborations. *Computer Networks*, v. 113, p. 176–187, 2016.

LI, H.; ZHANG, J.; TAN, W. Software defined network based protocol module multiplexing algorithm for wireless network. In: *IEEE Vehicular Technology Conference*. [S.l.: s.n.], 2016. v. 2016-July. ISBN 9781509016983. ISSN 15502252.

LI, Y.; PHAN, L. T. X.; LOO, B. T. Network functions virtualization with soft real-time guarantees. In: *Proceedings - IEEE INFOCOM*. [S.l.: s.n.], 2016. v. 2016-July. ISBN 9781467399531. ISSN 0743166X.

LIAO, Y.; YIN, D.; GAO, L. Network virtualization substrate with parallelized data plane. *Computer Communications*, v. 34, n. 13, p. 1549–1558, 2011. ISSN 01403664.

LIN, P.-C. P. C. et al. Balanced Service Chaining in Software-Defined Networks with Network Function Virtualization. *Computer*, v. 49, n. 11, p. 68–76, 2016. ISSN 00189162.

LOMBARDO, A. et al. An analytical tool for performance evaluation of software defined networking services. In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. [S.l.: s.n.], 2014. p. 1–7. ISBN 978-1-4799-0913-1.

MAI, H. L. et al. On the readiness of NDN for a secure deployment: The case of pending interest table. In: . [S.l.: s.n.], 2016. v. 9701, p. 98–110. ISBN 9783319398136. ISSN 16113349.

MAINI, E.; MANZALINI, A. Management and Orchestration of Virtualized Network Functions. In: *Monitoring and Securing Virtualized Networks and Services*. [S.l.]: Springer, 2014. p. 52–56.

MAINI, E. et al. A compositional modelling approach for live migration in Software Defined Networks. In: *2014 International Conference on the Network of the Future, NOF 2014 - Workshop on Smart Cloud Networks and Systems, SCNS 2014*. [S.l.: s.n.], 2014. p. 1–6. ISBN 9781479975310.

MAKHSOUS, S. H. et al. High available deployment of cloud-based virtualized network functions. In: *2016 International Conference on High Performance Computing & Simulation (HPCS)*. [S.l.: s.n.], 2016. p. 468–475. ISBN 978-1-5090-2088-1.

MAMATAS, L.; CLAYMAN, S.; GALIS, A. Information Exchange Management as a Service for Network Function Virtualization Environments. *IEEE Transactions on Network and Service Management*, v. 13, n. 3, p. 564–577, 2016. ISSN 19324537.

MANZALINI, A. et al. Clouds of virtual machines in edge networks. *IEEE Communications Magazine*, v. 51, n. 7, p. 63–70, 2013.

MAROTTA, M. A. et al. Resource sharing in heterogeneous cloud radio access networks. *IEEE Wireless Communications*, v. 22, n. 3, p. 74–82, June 2015. ISSN 1536-1284.

MARTINEZ, R. et al. Integrated SDN / NFV Orchestration for the Dynamic Deployment of Mobile Virtual Backhaul Networks over a Multi-layer ( Packet / Optical ) Aggregation Infrastructure. *Journal of Optical Communications and Networking*, v. 9, n. 2, p. A135–A142, 2017.

MARTINI, B.; PAGANELLI, F. A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks. *Future Internet*, v. 8, n. 2, 2016. ISSN 19995903.

MARTINS, J. et al. Clickos and the Art of Network Function Virtualization. In: USENIX ASSOCIATION. *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. [S.l.], 2014. p. 459–473.

MASSONET, P. et al. An architecture for securing federated cloud networks with service function chaining. In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. [S.l.: s.n.], 2016. p. 38–43.

MAYORAL, A. et al. The Need of a Transport API in 5G for Global Orchestration of Cloud and Networks through a Virtualised Infrastructure Manager and Planner. *Journal of Optical Communications and Networking*, X, n. 1, p. 1–7, 2016. ISSN 1943-0620.

MELO, R. et al. Virtual and unified address assignment for continuous communication in mobile networks. In: *IFIP Wireless Days*. [S.l.: s.n.], 2016. v. 2016-April. ISBN 9781509024940. ISSN 2156972X.

MENASCE, D. A. et al. *Performance by design: computer capacity planning by example*. [S.l.]: Prentice Hall Professional, 2004.

MENDELEY. 2018. Available from Internet: <https://www.elsevier.com/solutions/mendeley>.

MENG, S.; LIU, L.; WANG, T. State Monitoring in Cloud Datacenters. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 23, n. 9, p. 1328–1344, 2011.

MERKEL, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, Belltown Media, Houston, TX, v. 2014, n. 239, mar. 2014. ISSN 1075-3583. Available from Internet: <http://dl.acm.org/citation.cfm?id=2600239.2600241>.

MIJUMBI, R. et al. A connectionist approach to dynamic resource management for virtualised network functions. In: *2016 12th International Conference on Network and Service Management (CNSM)*. [S.l.: s.n.], 2016. p. 1–9.

MIJUMBI, R. et al. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, IEEE, 2015.

MIJUMBI, R. et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In: *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*. [S.l.: s.n.], 2015. p. 1–9. ISBN 9781479978991.

MONTAZEROLGHAEM, A. et al. A Load-Balanced Call Admission Controller for IMS Cloud Computing. *IEEE Transactions on Network and Service Management*, v. 4537, n. c, p. 806–822, 2016. ISSN 19324537.

MUHAMMAD, A. et al. Joint Optimization of Resource Allocation for Elastic Optical Intra-Datacenter Network. *IEEE Communications Letters*, IEEE, v. 20, n. 9, p. 1760–1763, 2016.

MUÑOZ, R. et al. Integrated SDN/NFV Management and Orchestration Architecture for Dynamic Deployment of Virtual SDN Control Instances for Virtual Tenant Networks [Invited]. *Journal of Optical Communications and Networking*, v. 7, n. 11, p. B62, 2015. ISSN 1943-0620.

NAIK, P.; SHAW, D. K.; VUTUKURU, M. Nfvperf: Online performance monitoring and bottleneck detection for nfv. In: IEEE. *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. [S.l.], 2016. p. 154–160.

NAKAUCHI, K.; NISHINAGA, N. Software-defined exchange for the virtualized WiFi network towards future Mobile Cloud services. In: *2016 IEEE International Conference on Communications Workshops, ICC 2016*. [S.l.: s.n.], 2016. p. 736–741. ISBN 9781509004485.

NAKAUCHI, K.; SHOJI, Y. VBS on the move: Migrating a virtual network for nomadic mobility in WiFi networks. In: *Proceedings - IEEE INFOCOM*. [S.l.: s.n.], 2016. v. 2016-Septe, p. 277–282. ISBN 9781467399555. ISSN 0743166X.

NANDUGUDI, A. et al. Network function virtualization: through the looking-glass. *Annales des Telecommunications/Annals of Telecommunications*, v. 71, n. 11-12, p. 573–581, 2016.

NAUDTS, B. et al. A dynamic pricing algorithm for a network of virtual resources. *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, v. 27, n. 2, p. 328–335, 2016. ISSN 10557148.

NEMATI, H. et al. Adaptive SLA-based elasticity management algorithms for a virtualized IP multimedia subsystem. In: *2014 IEEE Globecom Workshops, GC Wkshps 2014*. [S.l.: s.n.], 2014. p. 7–11. ISBN 9781479974702. ISSN 2166-0077.

NGUYEN, D. T. et al. Real- Time Optimized NFV Architecture for Internetworking WebRTC and IMS. In: *2016 17th International Telecommunications Network Strategy and Planning Symposium, Networks 2016 - Conference Proceedings*. [S.l.: s.n.], 2016. p. 81–88. ISBN 9781467389914.

NGUYEN, V.-G.; DO, T.-X.; KIM, Y. SDN and Virtualization-Based LTE Mobile Network Architectures: A Comprehensive Survey. *Wireless Personal Communications*, v. 86, n. 3, p. 1401–1438, 2016. Cited By 8.

NOMURA, K. et al. A System for Supporting Migration to Overlay OpenFlow Network Using OpenStack. In: *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*. [S.l.: s.n.], 2016. p. 595–598. ISBN 978-1-5090-0987-9.

PAOLINO, M. et al. SnabbSwitch user space virtual switch benchmark and performance optimization for NFV. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network, NFV-SDN 2015*. [S.l.: s.n.], 2015. p. 86–92. ISBN 9781467368841.

PASQUINI, R.; STADLER, R. Learning End-to-end Application QoS from OpenFlow Switch Statistics. In: *Netsoft 2017*. [S.l.: s.n.], 2017.

PFITSCHER, R. et al. DReAM - A Distributed Result-Aware Monitor for Network Functions Virtualization. In: *IEEE Symposium on Computers and Communication (ISCC)*. [S.l.: s.n.], 2016.

PFITSCHER, R. J. et al. A model for quantifying performance degradation in virtual network function service chains. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2018. p. 1–9. ISSN 2374-9709.

PFITSCHER, R. J.; PILLON, M. A.; OBELHEIRO, R. R. Diagnostico do provisionamento de recursos para maquinas virtuais em nuvens iaas. *31 Simposio Brasileiro de Redes de Computadores (SBRC)*, p. 599–612, 2013.

PFITSCHER, R. J.; PILLON, M. A.; OBELHEIRO, R. R. Customer-oriented Diagnosis of Memory Provisioning for IaaS Clouds. *ACM SIGOPS Operating Systems Review*, ACM, v. 48, n. 1, p. 2–10, 2014.

PHUNG-DUC, T. et al. Design and Analysis of Deadline and Budget Constrained Autoscaling (DBCA) Algorithm for 5G Mobile Networks. In: *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*. [S.l.: s.n.], 2017. p. 94–101.

PINHEIRO, B.; CERQUEIRA, E.; ABELEM, A. NVP: A network virtualization proxy for software defined networking. *International Journal of Computers, Communications and Control*, v. 11, n. 5, p. 697–708, 2016. ISSN 18419836 (ISSN).

PRADOS-GARZON, J. et al. Handover implementation in a 5G SDN-based mobile network architecture. In: *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. [S.l.: s.n.], 2016. p. 1–6. ISBN 978-1-5090-3254-9.

PRADOS-GARZON, J. et al. Modeling and Dimensioning of a Virtualized MME for 5G Mobile Networks. *IEEE Transactions on Vehicular Technology*, PP, n. 99, 2016. ISSN 00189545.

QUINTUANA RODRIGUEZ, V. K.; GUILLEMIN, F. Performance analysis of resource pooling for network function virtualization. In: *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*. [S.l.: s.n.], 2016. p. 158–163. ISBN 978-1-4673-8991-4.

RAHMAN, M. M.; DESPINS, C.; AFFES, S. Design Optimization of Wireless Access Virtualization Based on Cost & QoS Trade-Off Utility Maximization. *IEEE Transactions on Wireless Communications*, v. 15, n. 9, p. 6146–6162, 2016. ISSN 15361276.

REDHAT. *CVE-2018-5390*. [S.l.], 2018. Available from Internet: <https://access.redhat.com/articles/3553061#affected-products-2>.

RIEL, R. van. Measuring resource demand on linux. In: *Linux Symposium*. [S.l.: s.n.], 2006. p. 287.

RIGGIO, R. et al. Scheduling wireless virtual networks functions. *IEEE Transactions on Network and Service Management*, v. 13, n. 2, p. 240–252, 2016. ISSN 19324537.

RIGGIO, R.; RASHEED, T.; NARAYANAN, R. Virtual network functions orchestration in enterprise WLANs. In: *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*. [S.l.: s.n.], 2015. p. 1220–1225. ISBN 9783901882760. ISSN 1573-0077.

ROJAS-CESSA, R.; SALEHIN, K. M.; EGOH, K. Evaluation of switching performance of a virtual software router. In: *35th IEEE Sarnoff Symposium, SARNOFF 2012 - Conference Proceedings*. [S.l.: s.n.], 2012. ISBN 9781467314640. ISSN 19440367.

ROST, P. et al. Cloud technologies for flexible 5G radio access networks. *IEEE Communications Magazine*, IEEE, v. 52, n. 5, p. 68–76, 2014.

SAHHAF, S. et al. Network service chaining with efficient network function mapping based on service decompositions. In: *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. [S.l.: s.n.], 2015. p. 1–5.

SAHHAF, S. et al. Network service chaining with optimized network function embedding supporting service decompositions. *Computer Networks*, v. 93, p. 492 – 505, 2015. ISSN 1389-1286. Cloud Networking and Communications II. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S138912861500359X>.

SAPIO, A. et al. Per-user policy enforcement on mobile apps through network functions virtualization. In: *Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture - MobiArch '14*. [S.l.: s.n.], 2014. p. 37–42. ISBN 9781450330749.

SARIDIS, G. M. et al. Lightness: A Function-Virtualizable Software Defined Data Center Network with All-Optical Circuit/Packet Switching. *Journal of Lightwave Technology*, v. 34, n. 7, p. 1618–1627, 2016. ISSN 07338724.

SAUVANAUD, C. et al. Anomaly Detection and Root Cause Localization in Virtual Network Functions. In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. [S.l.: s.n.], 2016. p. 196–206. ISBN 978-1-4673-9002-6. ISSN 1542-1201.

SAVI, M.; TORNATORE, M.; VERTICALE, G. Impact of processing costs on service chain placement in network functions virtualization. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network, NFV-SDN 2015*. [S.l.: s.n.], 2015. p. 191–197. ISBN 9781467368841.

SCHOOLER, E. et al. *SIP: Session Initiation Protocol*. RFC Editor, 2002. RFC 3261. (Request for Comments, 3261). Available from Internet: <https://rfc-editor.org/rfc/rfc3261.txt>.

SCHULZ-ZANDER, J. et al. SecuSpot: Toward cloud-assisted secure multi-tenant WiFi hotspot infrastructures. In: *CAN 2016 - Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking, co-located with CoNEXT 2016*. [S.l.: s.n.], 2016. p. 61–66.

SCOPUS. 2018. Available from Internet: <https://www.scopus.com/>.

SHI, R. et al. MDP and Machine Learning-Based Cost-Optimization of Dynamic Resource Allocation for Network Function Virtualization. In: *Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015*. [S.l.: s.n.], 2015. p. 65–73. ISBN 9781467372817.

SIRACUSANO, G. et al. On the fly tcp acceleration with miniproxy. In: ACM. *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*. [S.l.], 2016. p. 44–49.

SUKSOMBOON, K.; FUKUSHIMA, M.; HAYASHI, M. Optimal virtualization of functionality for customer premise equipment. In: *IEEE International Conference on Communications*. [S.l.: s.n.], 2015. v. 2015-Septe, p. 5685–5690. ISBN 9781467364324. ISSN 15503607.

SUKSOMBOON, K. et al. Pending-interest-driven cache orchestration through network function virtualization. In: *2014 IEEE Global Communications Conference, GLOBECOM 2014*. [S.l.: s.n.], 2014. p. 1867–1872. ISBN 9781479935116.

SUKSOMBOON, K. et al. A dilated-cpu-consumption-based performance prediction for multi-core software routers. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. [S.l.: s.n.], 2016. p. 193–201.

SUKSOMBOON, K. et al. Towards performance prediction of multicore software routers. *International Journal of Network Management*, v. 27, n. 2, 2017.

TIRUMALA, A. et al. Iperf: The TCP/UDP bandwidth measurement tool. *htt p://dast. nlanr. net/Projects*, 2005.

TURCHETTI, R. C.; DUARTE, E. P. Implementation of Failure Detector Based on Network Function Virtualization. In: *Proceedings - 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2015*. [S.l.: s.n.], 2015. p. 19–25. ISBN 9781467380447.

TUTSCHKU, K. et al. On resource description capabilities of on-board tools for resource management in cloud networking and NFV infrastructures. In: *2016 IEEE International Conference on Communications Workshops, ICC 2016*. [S.l.: s.n.], 2016. p. 442–447. ISBN 9781509004485.

VASSILAKIS, V. G. et al. A software-defined architecture for next-generation cellular networks. In: *2016 IEEE International Conference on Communications, ICC 2016*. [S.l.: s.n.], 2016. ISBN 9781479966646. ISSN 15503607.

WANG, L. et al. Joint Optimization of Service Function Chaining and Resource Allocation in Network Function Virtualization. *IEEE Access*, v. 4, p. 8084–8094, 2016. ISSN 2169-3536.

WANG, M. A Streaming Data Anomaly Detection Analytic Engine for Mobile Network Management. In: *Proceedings - 13th IEEE International Conference on Ubiquitous Intelligence and Computing, 13th IEEE International Conference on Advanced and Trusted Computing, 16th IEEE International Conference on Scalable Computing and Communications, IEEE Internationa*. [S.l.: s.n.], 2016. p. 722–729.

WANG, Y.; ZHAO, Q.; ZHENG, D. Bottlenecks in production networks: An overview. *Journal of Systems Science and Systems Engineering*, Springer, v. 14, n. 3, p. 347–363, 2005.

WEN, T. et al. Network Function Consolidation in Service Function Chaining Orchestration. In: *2016 IEEE International Conference on Communications, ICC 2016*. [S.l.: s.n.], 2016. ISBN 9781479966646.

WERTHMANN, T.; GROB-LIPSKI, H.; PROEBSTER, M. Multiplexing gains achieved in pools of baseband computation units in 4G cellular networks. In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*. [S.l.: s.n.], 2013. p. 3328–3333. ISBN 9781467362351.

WU, J. et al. A Hierarchical Security Framework for Defending Against Sophisticated Attacks on Wireless Sensor Networks in Smart Cities. *IEEE Access*, v. 4, p. 416–424, 2016. ISSN 21693536.

WU, W.; HE, K.; AKELLA, A. Perfsight: Performance diagnosis for software dataplanes. In: *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. [S.l.: s.n.], 2015. p. 409–421.

XILOURIS, G. et al. T-NOVA: A marketplace for virtualized network functions. In: IEEE. *Networks and Communications (EuCNC), 2014 European Conference on*. [S.l.], 2014. p. 1–5.

XIONG, G. et al. Using the Cooperative Game for Service Placement of Virtual Network Functions. *China Communications*, v. 13, p. 146–157, 2016. ISSN 16735447.

XIONG, G. et al. An optimized deployment mechanism for virtual middleboxes in NFV-and SDN-enabling network. *KSII Transactions on Internet and Information Systems*, v. 10, n. 8, p. 3474–3497, 2016. ISSN 22881468.

YAMADA, K. et al. Design and Deployment of Enhanced VNode Infrastructure – Deeply Programmable Network Virtualization. *IEICE Transactions on Communications*, E99.B, n. 8, p. 1629–1637, 2016. ISSN 0916-8516.

YANG, H. et al. Performance evaluation of multi-stratum resources optimization with network functions virtualization for cloud-based radio over optical fiber networks. *Optics Express*, v. 24, n. 8, 2016.

YANG, H. et al. Performance evaluation of data center service localization based on virtual resource migration in software defined elastic optical network. *Optics Express*, v. 23, n. 18, p. 23059, 2015. ISSN 1094-4087.

YANG, H. et al. Performance evaluation of multi-stratum resources integration based on network function virtualization in software defined elastic data center optical interconnect. *Optics Express*, v. 23, n. 24, p. 31192–31205, 2015.

YANG, H. et al. C-RoFN: Multi-stratum resources optimization for cloud-based radio over optical fiber networks. *IEEE Communications Magazine*, v. 54, n. 8, p. 118–125, 2016. ISSN 01636804.

YI, B. et al. A comprehensive survey of network function virtualization. *Computer Networks*, Elsevier, 2018.

YOU, X. et al. A Coordinated Algorithm with Resource Evaluation for Service Function Chain Allocation. In: *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*. [S.l.: s.n.], 2016. p. 45–49. ISBN 978-1-5090-3936-4.

ZENG, M. et al. Orchestrating multicast-oriented NFV trees in inter-DC elastic optical networks. In: *2016 IEEE International Conference on Communications, ICC 2016*. [S.l.: s.n.], 2016. ISBN 9781479966646.

ZENG, M.; FANG, W.; ZHU, Z. Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks. *Journal of Lightwave Technology*, v. 34, n. 14, p. 3330–3341, 2016. ISSN 07338724.

ZHANG, B. et al. Toward online virtual network function placement in Software Defined Networks. In: *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. [S.l.: s.n.], 2016. p. 1–6. ISBN 978-1-5090-2634-0.

ZHANG, H. et al. User association scheme in Cloud-RAN based small cell network with wireless virtualization. In: *Proceedings - IEEE INFOCOM*. [S.l.: s.n.], 2015. v. 2015-Augus, p. 384–389. ISBN 9781467371315. ISSN 0743166X.

# APPENDIX A SCIENTIFIC PRODUCTION

The research work presented in this Thesis was reported to the scientific community through paper submissions to renowned conferences and journals. The process of doing research, submitting paper, gathering feedback, and improving the work helped to achieve the maturity hereby presented.

## A.1 Papers: accepted and on reviewing

The following list shows the main papers related to this Thesis.

1. Ricardo José Pfitscher, Eder John Scheid, Ricardo Luis dos Santos, Rafael Rodrigues Obelheiro, Mauricio Aronne Pillon, Alberto Egon Schaeffer-Filho, and Lisandro Zambenedetti Granville. **DReAM- A Distributed Result-Aware Monitor for Network Functions Virtualization**. 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 2016 pp. 663-668. DOI: 10.1109/ISCC.2016.7543813.
    - **Title:** *DReAM- A Distributed Result-Aware Monitor for Network Functions Virtualization*.
    - **Contribution:** An architecture to provide scalable and near real-time monitoring of NFV-enabled networks.
    - **Abstract:** In this paper, we describe DReAM's proposed architecture and its major components. We also discuss the feasibility of DReAM through experimental and analytical evaluations, where we observed application throughput, CPU utilization, communication overhead, scalability, and diagnosis complexity. We provide a trade-off analysis on the monitoring strategies in NFV scenarios. Our results indicate that a result-aware strategy is a better option when the monitored environment has more than 256 agents or when the diagnosis module induces at least 10% of CPU utilization.
    - **Status:** Published.
    - **Qualis:** A2.
    - **Conference:** Twenty-first IEEE Symposium on Computers and Communications.
    - **Date:** 27-30 June 2016
    - **Local:** Messina, Italy.
    - **URL:** <http://iscc2016.unime.it/>.
    - **Digital Object Identifier (DOI):** <http://dx.doi.org/10.1109/ISCC.2016.7543813>.
2. Ricardo José Pfitscher, Arthur Selle Jacobs, Eder John Scheid, Muriel Figueiredo Franco, Ricardo Luis dos Santos, Alberto Egon Schaeffer-Filho, Lisandro Zambenedetti Granville. **A Model for Quantifying Performance Degradation in Virtual Network Func-**

**tion Service Chains**. IEEE/IFIP Network Operations and Management Symposium (NOMS 2018), Taipei, Taiwan.

- **Title:** *A Model for Quantifying Performance Degradation in Virtual Network Function Service Chains*.
- **Contribution:** Modeling the amount of performance degradations imposed by each VNF of the service chain through a weighted sum of metrics.
- **Abstract:** In this paper, we introduce a model to quantify the `guiltiness` of a VNF on being a bottleneck in a service chain, which provides a metric that estimates the impact on processing delay. In addition, we propose an adaptive algorithm, based on linear regression and neural networks, to adjust the model parameters according to the environment particularities, such as the type and number of VNFs. We show through an experimental evaluation that the `guiltiness` metric faithfully characterizes end-service performance, by identifying up to 94% of the bottleneck VNFs in the analyzed scenarios. Also, we provide artifacts for researchers to reproduce our results in others scenarios.
- **Status:** Published.
- **Qualis:** A2.
- **Conference:** IEEE/IFIP Network Operations and Management Symposium (NOMS 2018).
- **Date:** 23-27 April 2018.
- **Local:** Taipei, Taiwan.
- **URL:** <http://noms2018.ieee-noms.org/>.
- **Digital Object Identifier (DOI):** <https://doi.org/10.1109/NOMS.2018.8406268>.

3. Ricardo José Pfitscher, Arthur Selle Jacobs, Luciano Zembruzki, Ricardo Luis dos Santos, Eder John Scheid, Muriel Figueredo Franco, Alberto Schaeffer-Filho, Lisandro Zambenedetti Granville. **Guiltiness: A Practical Approach for Quantifying Virtual Network Functions Performance**. Elsevier Computer Networks. *The International Journal of Computer and Telecommunications Networking*.

   - **Title:** *Guiltiness: A Practical Approach for Quantifying Virtual Network Functions Performance*.
   - **Contribution:** proposes a novel model for quantifying the performance of virtual network functions in general NFV scenarios.
   - **Abstract:** In this paper, we propose a model to quantify the `guiltiness` of each VNF on degrading the performance of a network service. This model consists of a novel application of the *Utilization Law* from queuing networks theory. Through numeric assessments on typical scenarios, we refined the set of relevant resource metrics and applied a weighted sum to gauge them in the model. Also, a hybrid

algorithm based on linear regression and neural networks is introduced to adjust the model's parameters according to the environment particularities, such as the type and number of VNFs in the service. Experimental evaluations confirm the ability of the model to (*i*) detect bottlenecks, and (*ii*) quantify performance degradations. When capacity restrictions were applied to different types of VNFs, our `guiltiness` metric was able to detect the root cause of degradations. Further, the `guiltiness` metric outperformed traditional metrics, being able to characterize 96.15% of the performance issues with a 25.6% reduction in the number of false positives when compared to CPU usage.

- **Status:** Submitted.
- **Qualis:** A1.
- **Journal:** Elsevier Computer Networks.
- **Date of submission:** October 2018.
- **URL:** <https://www.journals.elsevier.com/computer-networks>.
- **Digital Object Identifier (DOI):** <>.

4. Ricardo José Pfitscher, Arthur Selle Jacobs, Eder John Scheid, Muriel Figueiredo Franco, Ricardo Luis dos Santos, Alberto Egon Schaeffer-Filho, Lisandro Zambenedetti Granville. **Evaluation Scenarios in Network Function Virtualization Research: A Survey**. IEEE Communications Surveys & Tutorials.

    - **Title:** *Evaluation Scenarios in Network Function Virtualization Research: A Survey*.
    - **Contribution:** characterizes the current picture of NFV evaluation scenarios, classifying literature papers according to a taxonomy with four aspects: context of the research, performance evaluation, scenarios design, and service chain.
    - **Status:** to be submitted.
    - **Qualis:** A1.
    - **Journal:** IEEE Communications Surveys & Tutorials.
    - **Expected date of submission:** March 2019.
    - **URL:** <http://www.comsoc.org/cst>.
    - **Digital Object Identifier (DOI):** <>.

## A.2 Collaborations: accepted and on reviewing

The following list shows the main collaborations that, although not directly related to this thesis proposal, are linked to the design of network management solutions.

1. Muriel Figueredo Franco, Ricardo Luis dos Santos, Ricardo Cava, Eder John Scheid, Ricardo José Pfitscher, Carla Dal Sasso Freitas, Lisandro Zambenedetti Granville. **Interactive Visualizations for Planning and Strategic Business Decisions in NFV-Enabled Networks**. 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), pp. 492-499, 27-29 March 2017. Taipei, Taiwan. DOI:10.1109/AINA.2017.113
   - **Type:** Conference paper.
   - **Status:** Published.
   - **Qualis:** A2.
2. Ricardo Luis dos Santos, Muriel Figueredo Franco, Eder John Scheid, Ricardo José Pfitscher, Lisandro Zambenedetti Granville, Liane M Rockenbach Tarouco. **iMPROVE: Enhancing the Introduction of Services on Programmable Virtual Networks**. 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), pp. 500-507, 27-29 March 2017. Taipei, Taiwan. DOI:10.1109/AINA.2017.113
   - **Type:** Conference paper.
   - **Status:** Published.
   - **Qualis:** A2.
3. Arthur Selle Jacobs, Ricardo Luis do Santos, Muriel Figueredo Franco, Eder John Scheid, Ricardo José Pfitscher, Lisandro Zambenedetti Granville. **Affinity measurement for NFV-enabled networks: A criteria-based approach**. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 186-194, 8-12 May 2017. Lisbon, Portugal. DOI:10.23919/INM.2017.7987279
   - **Type:** Conference paper.
   - **Status:** Published.
   - **Qualis:** A2.
4. Arthur Selle Jacobs, Ricardo Luis do Santos, Muriel Figueredo Franco, Eder John Scheid, Ricardo José Pfitscher, Lisandro Zambenedetti Granville. **AMNESiA: Affinity measurement platform for NFV-enabled networks**. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 899-900, 8-12 May 2017. Lisbon, Portugal. DOI:10.23919/INM.2017.7987279
   - **Type:** Demo.

- **Status:** Published.
- **Qualis:** A2.

5. Eder J Scheid, Cristian Cleder Machado, Muriel F Franco, Ricardo Luis dos Santos, Ricardo José Pfitscher, Alberto Egon Schaeffer-Filho, Lisandro Zambenedetti Granville. **INSpIRE: Integrated NFV-based Intent Refinement Environment**. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 186-194, 8-12 May 2017. Lisbon, Portugal. DOI:10.23919/INM.2017.7987279

   - **Type:** Conference paper.
   - **Status:** Published.
   - **Qualis:** A2.

6. Arthur Selle Jacobs, Ricardo José Pfitscher, Ricardo Luis do Santos, Muriel Figueredo Franco, Eder John Scheid, Lisandro Zambenedetti Granville. **Artificial Neural Network Model to Predict Affinity for Virtual Network Functions**. IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 1-9, 2018. Taipei, Taiwan. DOI:10.1109/NOMS.2018.8406253

   - **Type:** Conference paper.
   - **Status:** Published.
   - **Qualis:** A2.

7. Arthur Selle Jacobs, Ricardo José Pfitscher, Ronaldo Alves Ferreira, Lisandro Zambenedetti Granville. **Refining Network Intents for Self-Driving Networks**. ACM SIGCOMM 2018 Afternoon Workshop on Self-Driving Networks (SelfDN 2018). Budapest, Hungary.

   - **Type:** Workshop paper.
   - **Status:** Published.
   - **Qualis:** None.

8. Arthur Selle Jacobs, Ricardo José Pfitscher, Ronaldo Alves Ferreira, Lisandro Zambenedetti Granville. **Refining Network Intents for Self-Driving Networks**. ACM SIGCOMM Computer Communication Review (CCR). November, 2018.

   - **Type:** Journal.
   - **Status:** To appear.
   - **Qualis:** B1.

9. Marcus Vinicius B Silva, Arthur Selle Jacobs, Ricardo José Pfitscher, Lisandro Zambenedetti Granville. **IDEAFIX: Identifying Elephant Flows in P4-Based IXP Networks**. IEEE Global Communications Conference (GLOBECOM 2018). Abu Dhabi, UAE.

- **Type:** Conference paper.
- **Status:** Published.
- **Qualis:** A1.

10. Lucas Bondan, Muriel Figueredo Franco, Leonardo da Cruz Marcuzzo, Giovanni Venâncio, Ricardo Luis dos Santos, Ricardo José Pfitscher, Eder John Scheid, Burkhard Stiller, Filip De Turck, Elias P. Duarte Jr., Alberto Egon Schaeffer-Filho, Carlos Raniery Paula dos Santos, Lisandro Zambenedetti Granville. **FENDE: Marketplace-based Distribution, Execution, and Lifecycle Management of VNFs**. *This paper was submitted to "Network & Service Management Series" of the IEEE Communications Magazine.*

    - **Type:** Journal.
    - **Status:** On Reviewing.
    - **Qualis:** A1.

# APPENDIX B VNFS DESCRIPTION

During the process of surveying the NFV literature several VNFs were found. Each VNF and their respective description are hereby presented.

## B.1 Security purpose VNFs

*DPI* – A deep packet inspection (DPI) is a network function that inspects each packet passing through it, logging details of network flows. As DPIs access the full stack of protocols, it also can function as a firewall, blocking and re-routing traffic according to policies (YI et al., 2018).

*Encryption/Decryption* – These network functions act to encrypt and decrypt communication data. Such NFs are placed between clients and end-services, commonly acting in non-secure networks (MASSONET et al., 2016).

*Failure detector* – A failure detector is a distributed oracle that provides information about the state of the processes of a distributed system. In general, the failure detector outputs the list of processes that are suspected to have crashed. (TURCHETTI; DUARTE, 2015)

*Firewall* – This network function acts as a barrier between internal and external networks, it generally is configured to define which kind of flow can income or outcome a network. In firewalls, policies apply through an access control list (ACL), establishing the addresses and ports allowed in the network (YI et al., 2018).

*IDS/IPS* – An intrusion detection system (IDS) monitor the network traffic to detect malicious activities. For doing this, the NF relies on historical observations of network traffic. When suspicious activity is detected, the NF informs a management entity to take actions upon it. The intrusion prevention system (IPS) is very similar to an IDS; the main difference is regarding the blocking action. While the IDS main role is to inform a manager about intrusions, the IPS first attempt to block the traffic to in sequence inform the manager about the occurrence (YI et al., 2018).

*Packet sniffer* – This network function aims to keep track of packets information in wireless networks. The functioning of a packet sniffer is quite similar to a DPI; the difference is that the usage of collected data also applies for transmission optimization, and not only for security purposes (RIGGIO; RASHEED; NARAYANAN, 2015).

*Proxy/Content filter* – According to the RFC3234 (BRIM; CARPENTER, 2002), a proxy (also known as content filter) is an intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. In fact, when a proxy is applied with security purposes it runs similarly to a firewall, blocking and allowing access to specific web sites.

*UCON* – The usage control (UCON) is a security VNF that prevents attacks on network. The

technology leverages continuous data control to take decisions regarding data access during user's session (WU et al., 2016).

*WAF* – A virtual web application firewall (WAF) is a specialized type of firewall. Such network function differentiates from regular implementations because of its ability to inspect the contents of HTTP communication (KRUEGER et al., 2010).

## B.2 Routing/Switching purpose VNFs

*L2/L3 forwarders* – These network functions appear in experimental evaluations where authors aim to implement generic VNFs that forwards packets according to information from protocols in layer two or three of the TCP/IP stack (HWANG; RAMAKRISHNAN; WOOD, 2015b).

*NFD* – A NDN forwarding daemon (NFD) is a network forwarder used in Named Data Networking (NDN) protocol implementation. The purpose of this VNF is to encapsulate all Interest and Data packets in IP/UDP channels used in the forwarding process (MAI et al., 2016).

*SDN controller* – In Software-Defined Networks (SDN), the controller entity is responsible for the network control plane logic. Regarding NFV, there are research works that argue in favor of the virtualization of the SDN controller (MUÑOZ et al., 2015). In the same context, a distributed flow switch controller (DFC) benefits for parallel processing for efficient processing of requests (DATE et al., 2014).

*Virtual switch/router* – This network function consists in a software-based implementation of switch devices. Similarly to traditional devices, virtual switches are in charge of receiving, processing, switching, and transmitting packets flowing in the network (GALLARDO; BAYNAT; BEGIN, 2016).

## B.3 Packet modification purpose VNFs

*NAT* – A network address translator (NAT) assigns the global address to hosts that use private IP address in the network. For the translation process, the VNF leverages an address table to perform global/local address conversions (YI et al., 2018).

*Tunneling* – According to the RFC3234 (BRIM; CARPENTER, 2002), tunnel endpoints, including virtual private network endpoints, use basic IP services to set up tunnels with their peer tunnel endpoints which might be anywhere in the Internet. Tunnels can also be used to wrap a packet inside another packet, which is the case of IPv6 tunneling over IPv4 packets (ADDIS et al., 2015).

## B.4 Performance purpose VNFs

*Load balancer* – The main objective of a load balancer is to reroute traffic between multiple paths/services according to demand of requests and availability of end nodes (YI et al., 2018).

*Monitor* – This network functions serve to the objective of continuously verifying the state of network services. By collecting key performance indicators from network devices and applications, such VNFs can provide meaningful diagnosis for network operators take actions accordingly (NAIK; SHAW; VUTUKURU, 2016).

*Proxy/Content cache* – Besides a proxy is denoted as a security network function, its primary design is tailored to performance purposes. As this network function substitutes the server/client during connections, it stores frequently accessed data in a cache to reduce the bandwidth consumption (YI et al., 2018).

*QoS/traffic shaper* – This type of VNF objectives to shape the amount of bandwidth used by flows in order to guarantee QoS policies (YI et al., 2018).

*Video/WAN optimezers* – Video and WAN optimization controllers (VOC and WOC) apply compression and caching techniques to reduce the latency of communication. Also, these functions can prioritize traffic streams according to configured policies (SAVI; TORNATORE; VERTICALE, 2015).

*TCP accelerator* – Such network functions are inserted in the middle of end-to-end communication. The approach consists in breaking TCP connections in multiple short TCP connections as many as the number of hops in the network. By doing such, it reduces the round-trip time of each connection, and consequently increases the overall throughput (KIM, 2015).

## B.5 RAN/EPC virtualization VNFs

*BRAS* – The Broadband Remote Access Server (BRAS) is a routing function of Internet service providers, it act to route traffic to and from broadband remote access devices. When inserted in the dataplane, it verifies the headers of packets, such as session numbers, PPPoE/PPP message types, and performs IP lookup and MAC header rewriting (MARTINS et al., 2014).

*BS* – The virtual base station (vBS) is a technology developed for continually provide connectivity for nomad terminals in WiFi networks. The idea consists in deploying service-specific vBS when a new target location is determined, maintaining the consistence of connections' state (NAKAUCHI; SHOJI, 2016).

*Duplicate filter* – This network function runs in wireless networks to filter out duplicate 802.11 frames based on their sequence number (RIGGIO; RASHEED; NARAYANAN, 2015).

*DNE-RTC control* – The dynamic network enabled RTC (DNE-RTC) is a VNF that acts to improve quality of service (QoS) of real time video calls by adjusting radio resources allocated to mobility users (JERO et al., 2016).

*Frame counter and filter* – Similarly to the duplicate filter, these network functions run in 802.11 networks to count and filter frames according to their sequence number (RIGGIO; RASHEED; NARAYANAN, 2015).

*HSS* – In the LTE envolved packet core (EPC) architecture, the home subscriber server (HSS) is the central user information database. Such function provides a mobile subscriber data repository, maintaining information about call and session setup, user authentication and access authorization (HAWILO et al., 2014).

*MME* – The Mobility Management Entity (MME) is the main signaling node in the EPC. This network functions acts for the authentication of mobile devices. It retains location information for each user and to define gateway routes (HAWILO et al., 2014). This VNF is also called as Mobility Management Function (MMF) in the paper published by (NAKAUCHI; SHOJI, 2016). In their work, Wen et al. (2016) also refers to as Management Access Gateway (MAG) a VNF resultant from the composition of MME and SGW.

*PCRF* – In the EPC architecture, the policy and charging roles function (PCRF) is responsible for passing and deciding the policies and charging in real time for each service and user (HAWILO et al., 2014).

*PGW* – The packet data network gateway (PDN-GW a.k.a, PGW) is the interconnection point between the EPC and external IP networks. In other words, this network function ensures connectivity between the user data plane and external networks (HAWILO et al., 2014). In their work, Wen et al. (2016) also refers to as Local Mobility Anchor (LMA) the PGW network function.

*PMIPv6* – The Proxy Mobile IPv6 (PMIPv6) is a function applied to support mobility management in LTE networks (WEN et al., 2016). Network-based mobility management enables hosts to have IP mobility. Where, the network is responsible for managing such IP mobility on behalf of the host (CHOWDHURY et al., 2008).

*Radio controller* – This network function applies in the control plane of software-defined radio access networks (SDRAN). In addition of defining routes for packets, as in SDN networks, such function adjusts wireless communication parameters (YANG et al., 2016).

*SGW* – The serving gateway (SGW) is another important component of the EPC. Such function runs in the user plane, and its action consists in routing and forwarding data packets from/to base stations, being logically connected to the PGW (HAWILO et al., 2014).

*SRV* – According to the RFC2782 (GULBRANDSEN; ESIBOV, 2000), the service record (SRV) is a network function used for dynamic server selection in domain name service (DNS) lookups. The main objective of using SRVs is to provide load balancing for pool

of DNS servers.

*WGW* – Rahman, Despins and Affes (2016) uses the term WGW when referring to a delay-tolerant traffic service. However, authors neither describe the functioning of this VNF nor provide additional reference to it.

## B.6 IMS internal components VNFs

In Chapter 3, Section 3.2.2, we describe the Clearwater implementation of IMS internal components, in this Appendix we discuss the core functions of the IMS architecture (CAMARILLO; GARCIA-MARTIN, 2007).

*CSCF* – In the IMS, three Call Session Control Functions (CSCF) act in the processing of SIP signaling packets. These components' roles include call initiation, proxy, and session serving.

- *I-CSCF* – The interrogating-CSCF (I-CSCF) is the function that assigns user equipments to the appropriate server. By doing such, this component acts as an internal proxy, intermediating the traffic between P-CSCF and S-CSCF components.
- *P-CSCF* – The proxy-CSCF (P-CSCF) is the entry point for users that contact an IMS, as resigned in its name, this function acts as a proxy, receiving, processing and forwarding all the SIP traffic to and from user equipments. As the function is the first connection point, it can also provide SIP answers for users, and relieve sessions when abnormal behavior occurs.
- *S-CSCF* – The serving-CSCF (S-CSCF) is the network function that acts as the SIP server, and performs session control. Such IMS component is also responsible for deciding whether or not SIP messages can be forwarded to application servers.

*HSS* – As in the EPC, in IMS implementations, the home subscriber server (HSS) is a user database that contains subscription-related information. Such a function is responsible for performing authentication and authorization of mobile users and providing the IP and localization about subscribers.

*MGW* – The media gateway (MGW) function acts as an interface between SIP-based IMS networks and traditional PSTN networks.

*SIP* – The session initiation protocol (SIP), an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants (SCHOOLER et al., 2002). Despite the acronym SIP refers to a protocol, it also appears in papers' evaluation when relating to SIP clients or servers.

## B.7 Other purpose VNFs

*Address assigner* – Melo et al. (2016) propose a novel schema for address assignment in mobile networks. The VNF functioning divides into two objectives: provide a method to unify address formats, and define how to assigning device addresses in heterogeneous networks.

*Configuration acquiring function* – Such VNF applies in the management of software-defined networks. The network function acts for collecting configurations (*e.g.*, VLAN tags, connecting interfaces, and Openflow roles) of Open vSwitches and VMs (NOMURA et al., 2016).

*DNS* – The domain name system (DNS) is one of the most common network services. The DNS is a service that resolves domain names in IP addresses. A domain name is a mechanism that names resources usable in different hosts, networks, protocol families, internets, and administrative organizations (IETF, 1987).

*MAI* – The Mobile App Identification (MAI) is a network function that leverage traffic information to categorize mobile application. The application categories, and traffic can be further used in network policy enforcements (SAPIO et al., 2014).

*MiMP* – A Man-in-the-middle Proxy (MiMP) is proposed by (SAPIO et al., 2014). Such function acts for enforcing policies on mobile applications. In its functioning, the VNF extracts information from encrypted and non-encrypted users' traffic to give them to management entities.

*NVP* – The network virtualization proxy (NVP) is a VNF that resides between virtual switches and controllers in SDN-enabled networks. As with general proxies, the primary aim of NVP is to improve the performance of communication among switches and controllers (PINHEIRO; CERQUEIRA; ABELEM, 2016).

*PCE* – A path computation element (PCE) is a network function responsible for determining best routes between sources and destinations in optical networks. When virtualized, such network function allows the on-demand path computation virtual optical networks requests (YANG et al., 2015b).

*STB functions* – A set-top box (STB) is an equipment that runs in home environments to provide user entertainment functions, such as IPTV transmissions and multimedia applications (*e.g.*, Skype, Netflix, and Spotify) (SUKSOMBOON; FUKUSHIMA; HAYASHI, 2015).